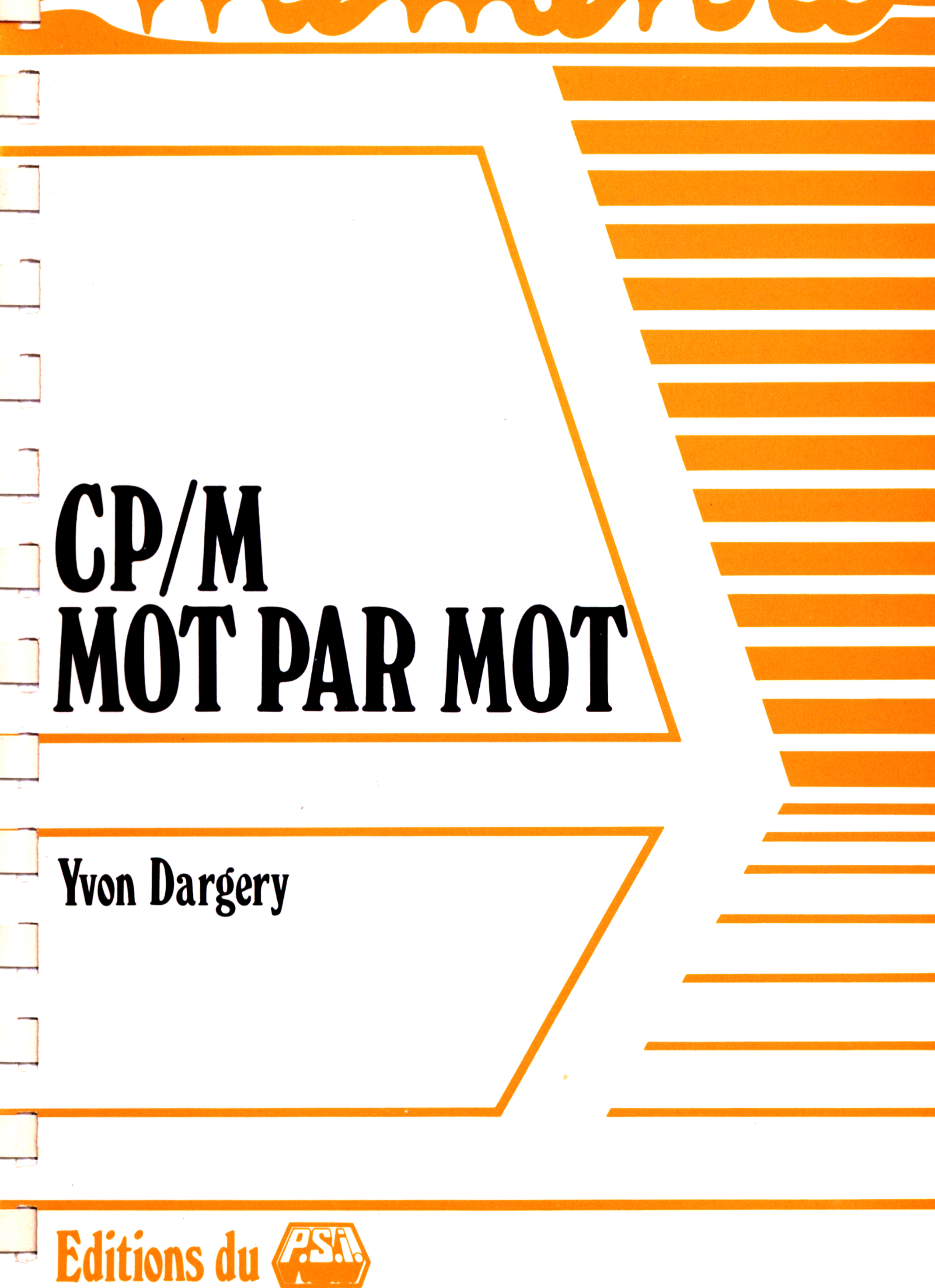
mémento



CP/M MOT PAR MOT

Yvon Dargery

Editions du P.S.I.

CP/M MOT PAR MOT

La collection « **MEMENTOS** » est constituée de recueils de données sur un type de matériels. Dix titres sont actuellement disponibles dans cette collection :

— Clefs pour le TRS-80 — tomes 1 et 2 — Rémy Pineau
— Le Basic de A à Z — Jacques Boisgontier
— Clefs pour l'Apple II — Nicole Bréaud-Pouliquen
— CP/M mot par mot — Yvon Dargery
— Clefs pour le ZX-81 — Jean-François Séhan
— Clefs pour le ZX-spectrum — Jean-François Séhan
— Clefs pour le CBM — Daniel-Jean David
— Clefs pour le VIC — Daniel-Jean David
— PC/DOS mot par mot — Yvon Dargery

RAPPELS

Les séries :

En fait, il faudrait parler de niveaux, puisque la couleur attachée à chaque ouvrage permet de situer la « force » de celui-ci selon le code suivant :

Série VERTE : ouvrage d'initiation ne nécessitant que des connaissances de base.
Série BLEUE : suppose une connaissance élémentaire du sujet traité.
Série ROUGE : ouvrage d'approfondissement, niveau de complexité moyen.
Série NOIRE : ouvrage d'approfondissement, niveau de complexité élevé.

Les collections :

Les ouvrages d'Edition du PSI sont répartis en collections :

« **LANGAGES** », « **MATERIELS** », « **PROGRAMMES** », « **GUIDES PRATIQUES** », « **MEMENTOS** », « **UTILISATIONS DE L'ORDINATEUR** », « **LOGIGUIDE** » et pour l'initiation, outre quelques livres hors collection, « **... POUR TOUS** ».

ISBN : 2-86595-072-7

CP/M MOT PAR MOT

par
Yvon Dargery



Editions du P.S.I.
1984

SOMMAIRE

AVANT-PROPOS

CP/M (Contrôle de Processus pour Microprocesseurs ou l'équivalent en anglais) est le nom d'un système d'exploitation de disquettes, c'est-à-dire un logiciel qui permet à l'ordinateur de s'utiliser de manière plus simple par rapport surtout à l'utilisation des disques (place disponible, nom des fichiers...).

Il est certain que **CP/M** est devenu de fait le standard des systèmes d'exploitation de disquettes pour les ordinateurs bâtis autour d'un microprocesseur 8 bits.

Conçu à l'origine pour le processeur 8Ø8Ø, on a pris l'habitude de parler du **CP/M 8Ø** pour parler de ce système d'exploitation.

L'avènement des ordinateurs bâtis autour d'un microprocesseur 16 bits (comme l'IBM PC) a remis en cause la suprématie de CP/M comme système d'exploitation : un autre concurrent se présente sur le créneau MS DOS.

Digital Research, créateur de **CP/M 8Ø** ne s'avoue pas vaincu et propose son CP/M 86, adaptation du **CP/M 8Ø** au microprocesseur 8Ø86, le plus répandu dans les ordinateurs 16 bits.

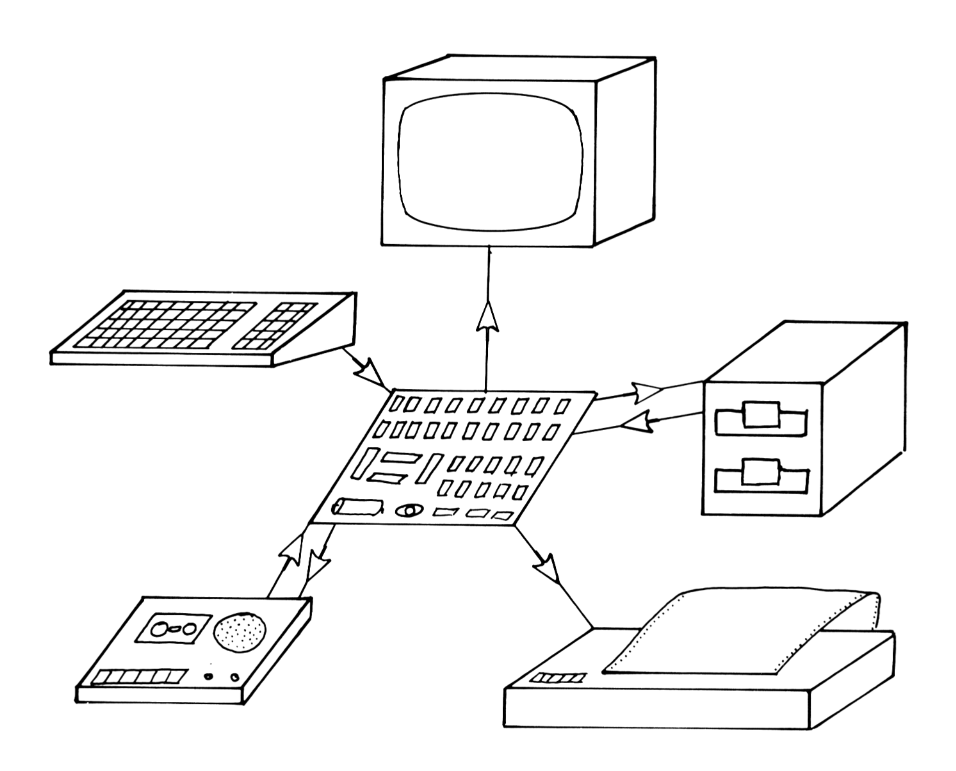
La part du marché entre ces deux systèmes d'exploitation varie de mois en mois, une chose est certaine, à eux deux, ils se partagent la quasi totalité des ordinateurs.

DIFFERENCES ENTRE CP/M 8Ø ET CP/M 86

Si les différences structurelles et fonctionnelles sont importantes, pour l'utilisateur elles sont minimes ; les ordres et la syntaxe de **CP/M 8Ø** se retrouvent pour l'essentiel dans CP/M 86.

Quelques ordres ont été ajoutés, surtout, d'une machine à l'autre, quelques ordres nouveaux marquent la volonté des constructeurs de se démarquer en ajoutant des facilités nouvelles.

Dans le texte, les ordres communs seront indiqués sans signal particulier, les ordres CP/M 86 seront signalés comme spécifiques.



CP/M doit assurer la gestion d'accès à tous les périphériques. Il se comporte comme le programme prioritaire de fonctionnement matériel de l'ordinateur

RAPPELS

Ces points sont importants, ils sont nécessaires à la compréhension de la suite de notre propos, dans un premier temps cependant, vous pourrez vous contenter de vous référer aux mots-clé quand vous serez gêné dans le reste de votre lecture.

ACTIF (LECTEUR ACTIF)

On parle souvent du lecteur actif sous CP/M ; il s'agit de celui qui est en rapport direct avec la machine (plus prosaïquement, celui qui tourne).

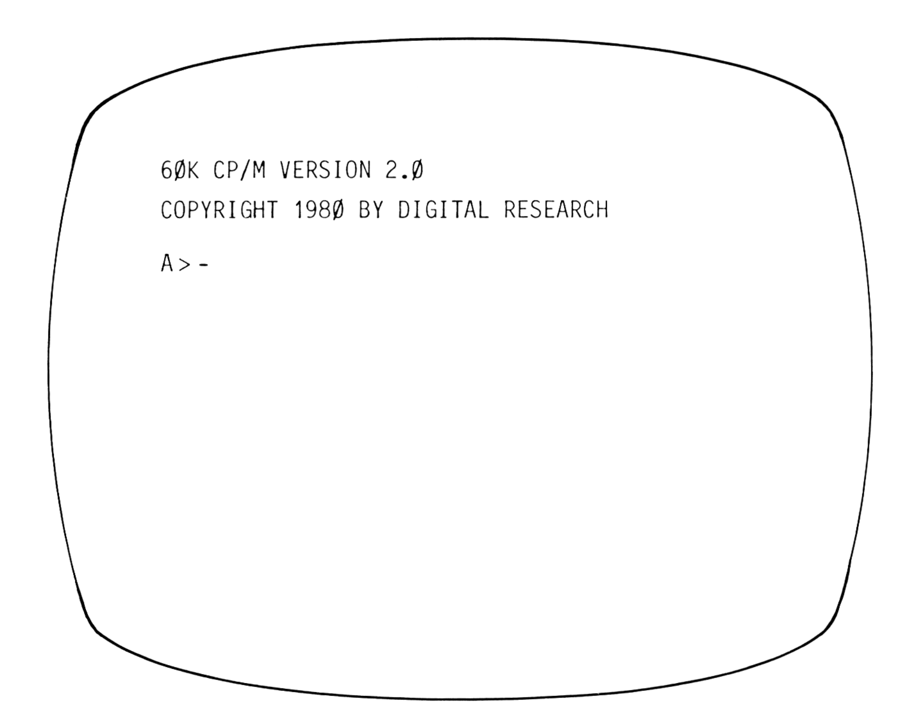
Lors du départ à froid, c'est le lecteur A qui est en général actif, (celui dans lequel on met le disque système) pour rendre le lecteur B actif, il suffit de taper B :

BOOT

Encore appelé **BOOTSTRAP**, c'est une manière bien peu française de nommer la procédure de chargement du programme CP/M en mémoire vive. En effet, comme nous l'avons vu, CP/M est un programme de gestion des organes périphériques et notamment des disques. La première opération consiste donc à l'implanter depuis le disque en mémoire vive.

Nous parlerons couramment de **démarrage** ou de **départ**, ces termes sont plus explicites pour nous ; le premier démarrage, celui qui charge CP/M lors de la mise sous tension de la machine s'appelle le **démarrage à froid.**

Puisque CP/M garde en mémoire vive la liste des fichiers qui existent sur le disque, si on change le disque sans le lui faire savoir, il est perdu et nous le fait savoir sans politesse (message de la forme BDOS ERROR). Pour lui dire que l'on a changé de disque, on fait un **démarrage à chaud.**



L'affichage à l'écran dès la mise en service de CP/M
Le A>- signale que l'unité de disquette A est active

Cette opération s'effectue en appuyant sur **CONTROLE C** (touche **BREAK** sur de nombreux systèmes) dès que la trappe contenant le nouveau disque a été fermée. CP/M ajuste alors sa mémoire vive avec le contenu du disque.

L'opération de **démarrage** à chaud ou à froid est indispensable sous CP/M ; l'oublier expose l'utilisateur à des ennuis lors de tentatives d'écriture sur le nouveau disque.

COMMANDES

Sous CP/M, les commandes sont les ordres directement compréhensibles par le système standard.

On distingue deux sortes de commandes :

- les **commandes résidentes** font partie de CP/M proprement dit, elles résident dans le système dès qu'il est chargé : il s'agit des commandes DIR, ERA, REN, SAVE, TYPE, USER...

- les **commandes temporaires** sont des commandes de gestion du système, livrées en standard par le fournisseur mais qui résident sur disque et qui sont chargées en mémoire vive pour le temps de l'utilisation : il s'agit des commandes PIP, STAT, SYSGEN, SUBMIT... des commandes liées à l'éditeur ou au débugger : ED, ASM, DDT... de celles liées à votre ordinateur comme le formatteur ou l'utilitaire de recopie de disquettes...

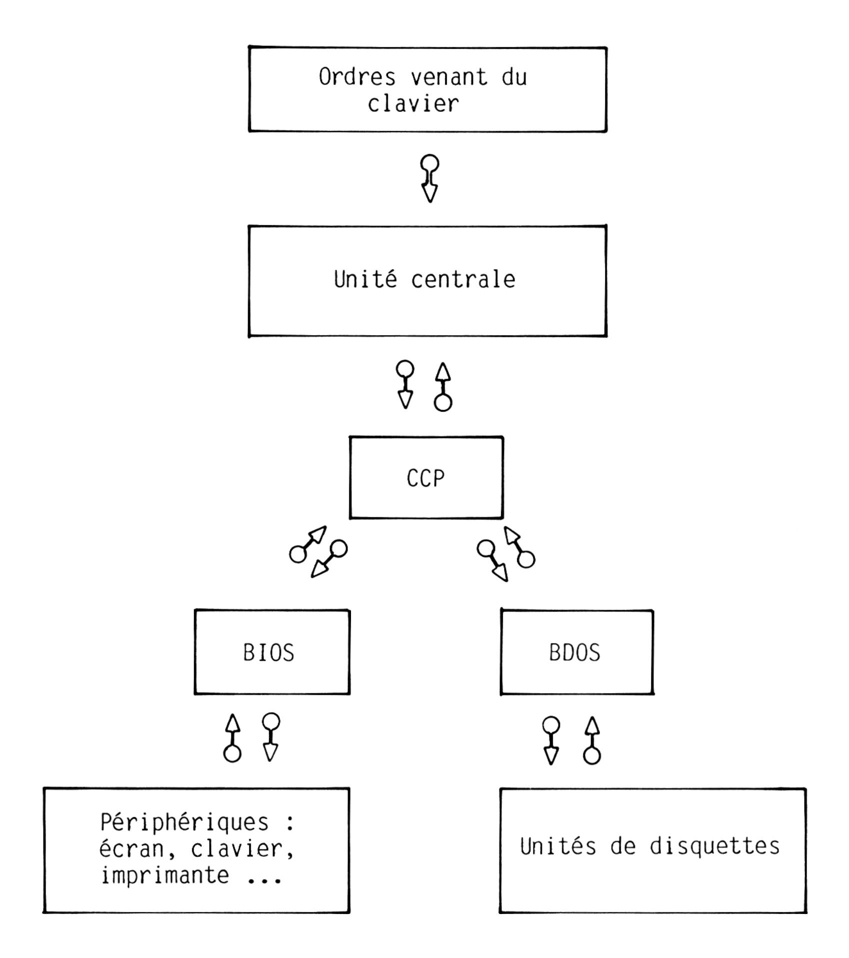
Les commandes temporaires figurent comme des fichiers avec l'extension COM.

CONTROLE (TOUCHE CONTROLE)

Votre clavier possède une touche appelée **CONTROLE** ou **CTRL.** Cette touche est essentielle à l'utilisation de CP/M ; elle a pour rôle de modifier le code des touches du clavier et donc à créer une autre signification à la touche frappée.

Par exemple, la touche **c** donne un **c**, la même touche tapée en appuyant en même temps sur la touche **SHIFT** donne la lettre **C majuscule**, la même touche tapée en même temps que la touche **CTRL** a le même rôle que la touche **BREAK** (on parle de la commande "Control C").

De nombreuses touches ont ainsi un rôle défini standard, mais la quasi totalité d'entre elles ont un rôle dans certains programmes d'application.



CP/M gère l'environnement de l'ordinateur à travers trois programmes spécialisés : CCP-BIOS-BDOS

Dans la suite, nous utiliserons couramment le signe ↑ pour contrôle, **↑C signifiera donc contrôle C.**

Voici les commandes contrôlées les plus usuelles :

CTRL C : Interrompt l'opération en cours et rend la main à CP/M.
CTRL E : Crée un retour à la ligne.
CTRL H : Retour en arrière du curseur d'une position de lettre.
CTRL J : Saut d'une ligne.
CTRL P : Echo imprimante. Si une imprimante est branchée, tout texte visualisé à l'écran est frappé à l'imprimante en même temps.
CTRL R : Redonne la ligne en cours de correction dans son état corrigé.
CTRL S : Suspend l'affichage d'un texte à l'écran, par exemple un second CTRL S reprend l'affichage.
CTRL Z : Marque la fin d'une opération (fin d'une ligne de texte par exemple).

CP/M

Le système d'exploitation est constitué en fait de plusieurs parties, en ce qui concerne la position dans la mémoire mais aussi en termes de fonctions.

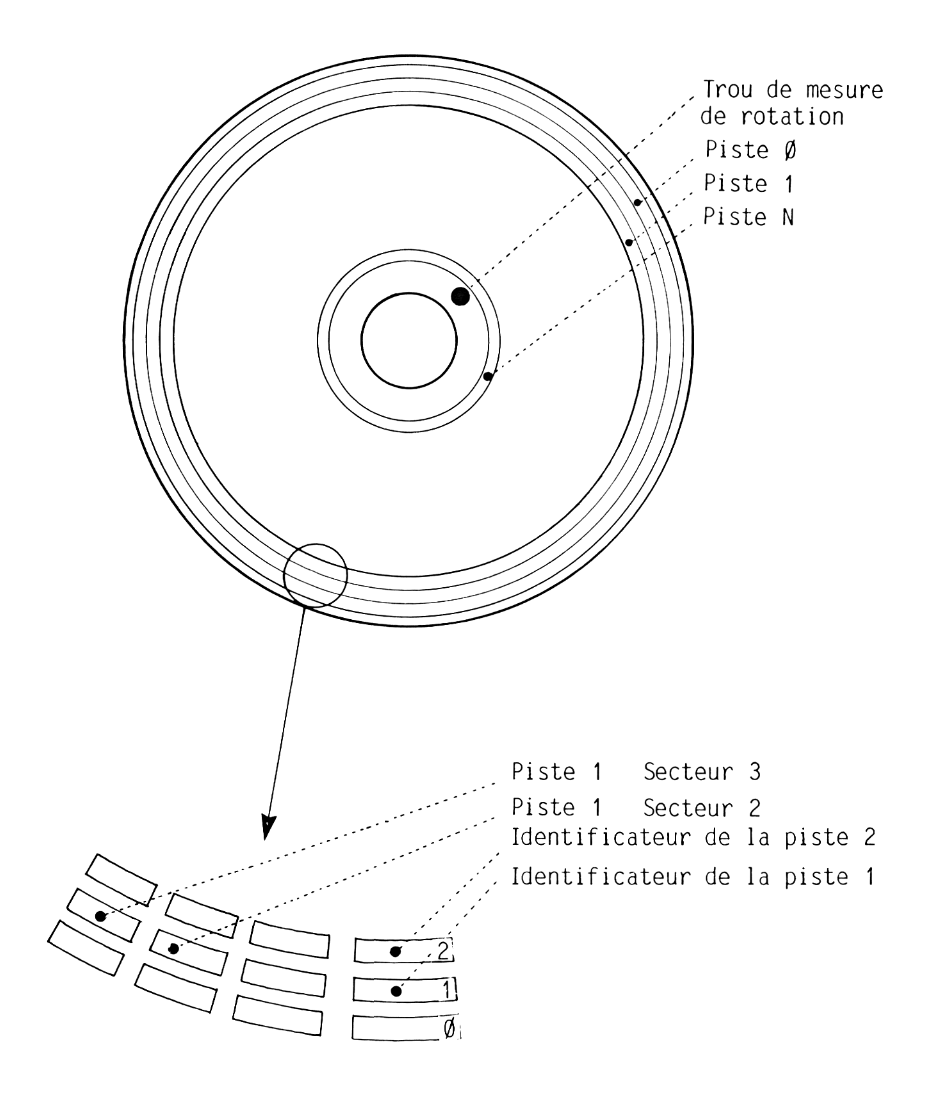
Voici ces zones classées dans l'ordre de leur apparition sur la carte mémoire :

- *la zone du système (ou PAGE Ø) :* Elle n'appartient pas exactement à CP/M, elle caractérise l'ordinateur hôte. Elle contient notamment les utilitaires de départ avant le chargement de CP/M ;

- *la ZONE TPA (pour Transian Program Area, zone de programmes transitoire) :* c'est la zone laissée accessible à l'utilisateur. Ici seront écrits les programmes, les utilitaires, les données qui n'appartiennent pas explicitement à CP/M ;

- *la ZONE CCP (pour Console Command Processor, contrôle des commandes de l'opérateur) :* Elle est chargée de gérer les commandes qui sont envoyées par l'opérateur au moyen de la console (écran-clavier, en l'occurrence du clavier) ; les commandes reconnues par CCP sont les **"commandes résidentes"** de CP/M que nous verrons plus loin.
Cette zone est immédiatement placée après la zone utilisateur, il peut donc arriver qu'elle soit détruite si la place est insuffisante. Dans ce cas, les ordres CP/M ne seront pas reconnus lors du retour au système et un démarrage à chaud sera nécessaire ;

- *la ZONE BDOS (pour Basic Disk Operating System) :* Elle contient les programmes propres de gestion de la lecture et de l'écriture



Les données sont systématiquement organisées pour que CP/M puisse y accéder à partir d'un schéma de classement

des fichiers sur disque, elle est indispensable au système et ne peut donc pas être détruite même en cas de manque de place en mémoire centrale ;

- *la ZONE BIOS (pour Basic Input Output System) :* C'est celle qui permet à tel ou tel système ou à telle ou telle configuration de tourner sous CP/M ; elle est adaptée par le constructeur en fonction des caractéristiques propres de sa machine (type d'écran, de clavier, de périphériques spécifiques...).

C'est cette zone que vous devrez modifier si vous achetez un CP/M standard et que vous souhaitez le faire tourner... Je vous souhaite alors bien du plaisir si vous connaissez mal les caractéristiques des périphériques que vous aurez à utiliser ou si vous n'êtes pas né avec un couple d'ordinateurs comme parents.

Les deux dernières zones (*BDOS* et *BIOS)* sont appelées *FDOS (Functional Disk Operating System)*, c'est-à-dire l'ensemble des ordres de gestion du système.

Les trois dernières zones (*CCP, BDOS, BIOS)* forment le CP/M proprement dit.

DENSITE

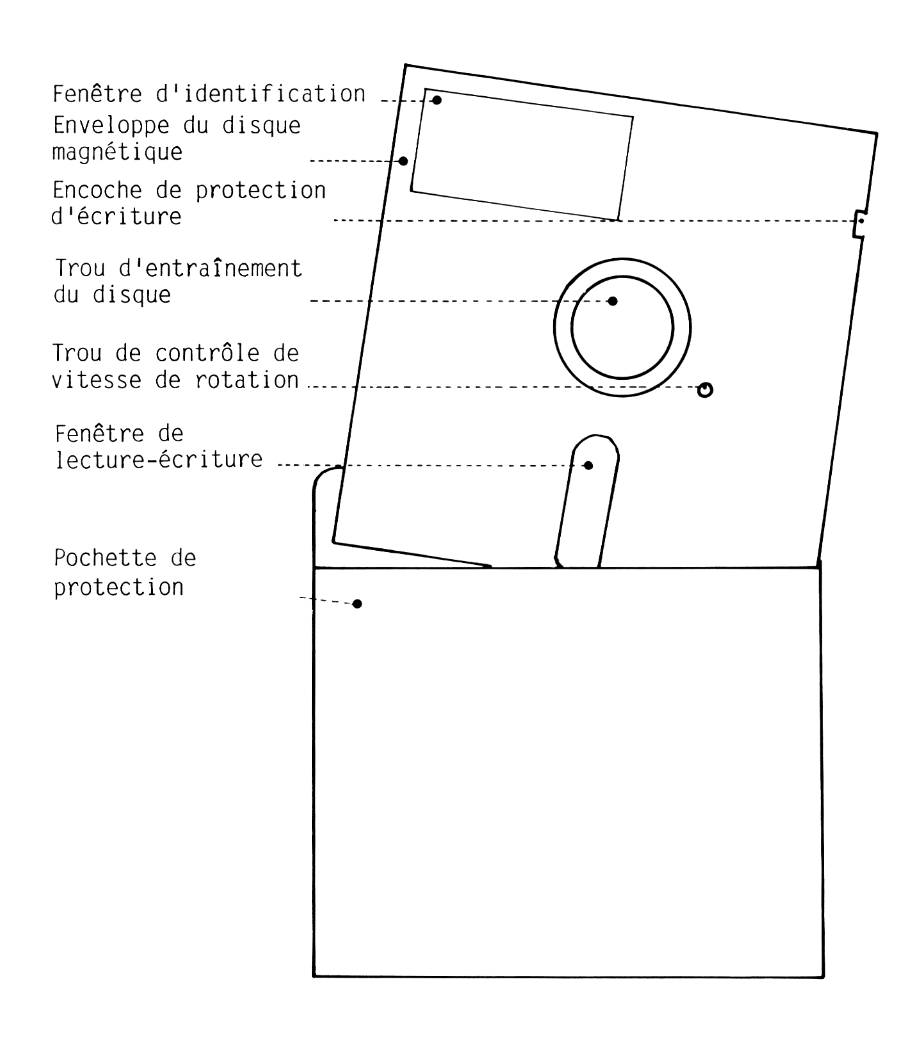
La densité des informations enregistrées sur une disquette caractérise la capacité de celle-ci. Sur une disquette de 5 pouces ¼ qui épuipe le plus grand nombre des petits systèmes tournant sous CP/M, les densités vont de 1ØØ kilo octets à 1 méga octet par disquette, les différences constatées sont dues à trois facteurs :

- *le nombre de faces de la disquette :* en simple face, une seule tête magnétique est en contact avec le disque alors qu'en double face, deux têtes lisent simultanément la surface magnétisée.

- *la densité des informations :* elle est fonction de la vitesse de rotation. Si la disquette tourne deux fois moins vite, on pourra enregistrer deux fois plus d'informations sur la même surface. Dans ce cas, bien sûr, l'électronique devra être plus sophistiquée.

- *le type de piste :* il est lié à la largeur de la tête de lecture; en double piste, on enregistrera donc deux fois plus d'informations. Là encore, l'éléctronique doit compenser par une augmentation de qualité la perte d'informations due à la réduction physique du support.

Un dernier perfectionnement est celui de la vitesse différentielle de rotation qui fait tourner moins vite le disque au fur et à mesure de son passage en périphérique de façon à avoir la même surface lue dans la même unité de temps. Cette possibilité,


Fenêtre d'identification
Enveloppe du disque magnétique
Encoche de protection d'écriture
Trou d'entraînement du disque
Trou de contrôle de vitesse de rotation
Fenêtre de lecture-écriture
Pochette de protection

liée aux progrés de la microélectronique n'existe que sur peu de machines.

On parlera donc de système en simple face/simple densité/simple piste ou en double face/double densité/double piste avec toutes les combinaisons.

Il est bien évident que la technologie de l'ordinateur ne peut pas tout et que plus le lecteur de disquette sera sophistiqué et plus les supports employés devront être adaptés et donc de meilleure qualité.

DIRECTORY

Appelé CATALOGUE en France, c'est le répertoire des fichiers connus de CP/M et enregistrés sur la disquette, il contient notamment le nom des fichiers, le repère de début et le repère de fin.

Le système est donc capable de retrouver les fichiers mais aussi d'en connaître leur encombrement. Ce catalogue est chargé en partie en mémoire vive lors du BOOT. Pour toute nouvelle écriture, il devra y avoir similitude entre les éléments en mémoire et ceux qui figurent sur la disquette d'où l'interdiction de changer de disquette d'un lecteur sans faire un démarrage à chaud (Contrôle C sous CP/M, RESET sous MBASIC).

DISQUETTE

C'est le support magnétique de l'information. Elle est constituée d'une pochette revêtue intérieurement de téflon et dans laquelle tourne librement un disque de polyester recouvert de particules magnétisables comme pour une bande magnétique de magnétophone.

La pochette possède :

- un trou central d'entraînement,
- un petit trou excentré de contrôle de la vitesse de rotation (un trou similaire est percé dans le disque, une cellule photoélectrique lit la rotation),
- une encoche de verrouillage en écriture.

La disquette est un support fragile et précieux par les informations qu'il contient, il est donc nécessaire de le manipuler avec beaucoup de précautions (voir les consignes d'utilisation sur la pochette) et ne pas hésiter à investir dans les disquettes de meilleure qualité.

DOS

Employé pour **Disk Operating System**, les français préfèrent parler de **SED** pour **Système** d'**Exploitation de Disques.** Il s'agit de toutes les procédures automatisées pour utiliser au mieux l'espace de stockage des supports magnétiques sans encombrer l'opérateur par des ordres complexes que nécessiteraient les opérations entièrement manuelles (point de placement de la tête, liste des fichiers, place disponible etc...).

Le DOS fonctionne donc comme un gestionnaire de l'espace disque, il doit connaître la place prise par chaque information et pouvoir utiliser au mieux la place restante.

FICHIERS

Pour CP/M, un fichier est un ensemble d'informations stockées sur disque sous un nom particulier et répertorié au directory. Un fichier peut donc contenir indifféremment des programmes, des données, des fiches...

Nom des fichiers

Un fichier est caractérisé par trois éléments :

- *le nom du lecteur* caractérisé par une lettre, suivi de deux points (A: ou B: ou D: ...),
- *le nom du fichier* composé de n'importe quel caractère alphanumérique, de moins de 8 signes et ne contenant pas d'espace, et en général de signe autre que les chiffres et les lettres.
  Un point sépare le nom du fichier de son type, caractérisé par trois lettres ou chiffres. Dans certains cas, le type est mis par défaut par CP/M, par exemple, un programme sauvé sous Basic, sans extension explicite sera sauvé avec l'extension BAS.
- *l'extension* fait partie du nom du fichier, c'est un mot de trois lettres, séparé du nom du fichier par un point. Cette extension peut être imposée par l'utilisateur ou faire défaut et selon le cas, le système placera une extension ou n'en mettra pas, par exemple, sous Basic, si je tape SAVE "PROGR.ØØ1", l'extension sera ØØ1 ; si je tape SAVE "PROGR", le système sauvera le fichier PROGR.BAS., l'extension BAS étant mise par défaut puisque le système travaille sous Basic ; les extensions standards les plus courantes sont :

COM : pour les programmes écrits en assembleur, directement opérationnels et accessibles sous CP/M.

ASM : pour les fichiers source utilisés avec la commande ASM.

HEX : pour les fichiers en langage machine en format hexadécimal, prêts à être chargés par l'ordre LOAD.

BAK : pour les fichiers source d'une édition sous éditeur (sauvegarde précédente).

$$$: pour les fichiers temporaires de l'éditeur ED.

SUB : pour les fichiers d'ordres exécutables par un programme SUBMIT.

D'autres extensions par défaut existent, elles sont souvent caractéristiques d'un langage, vous les découvrirez si vous utilisez un de ceux-ci.

Exemple

A:FICHIER.ØØ1	est correct.
B:PROG1	est correct. Il n'y a pas de type, celui-ci est optionnel.
PROG1.BAS	est correct, en l'absence du nom du lecteur, l'ordinateur choisira le lecteur actif pour écrire ou pour lire.
PROGRAMME1.BAS	est incorrect (1Ø signes dans le nom).
PRO.FIC.BAS	est incorrect (un point dans le nom).

Noms des fichiers en CP/M 86

Ils sont semblables aux noms de fichiers CP/M 8Ø, quelques différences au niveau des extensions :

- les fichiers exécutables ont l'extension .COM en CP/M 8Ø, ils ont l'extension .CMD en CP/M 86 ;
- les fichiers assembleur ont l'extension .ASM en CP/M 8Ø, ils ont l'extension A86 en CP/M 86 ;
- les fichiers hexa ont l'extension .HEX en CP/M 8Ø, ils ont l'extension .H86 en CP/M 86...

En **CP/M 86**, les 4 unités logiques ont pour nom :

CON : pour le clavier/écran
AXI : pour l'entrée auxiliaire
AXO : pour la sortie auxiliaire
LST : pour la sortie imprimante

FORMATTAGE

Une disquette est un support vierge, aucun repère ne permet à l'ordinateur de savoir s'il se trouve sur la piste 1, 2 ou 25... ni même s'il se trouve sur une piste ou entre deux. La disquette doit donc être **formattée**, c'est-à-dire qu'elle doit posséder les repères nécessaires. Ces repères représentant les pistes et les secteurs, la machine saura alors sur quel numéro de piste elle se trouve et par conséquent, connaissant le catalogue, si on lui demande de charger tel ou tel programme, elle saura sur quelle piste elle doit aller le chercher.

Il y a deux types de formattage, le **formattage Hard** (matériel) et le **formattage Soft** (logiciel).

Dans le **formattage Hard**, la disquette possède plusieurs trous en regard des deux trous excentrés de la pochette.
La matérialisation des secteurs est donnée par le signal de passage des trous de la disquette devant le trou de la pochette.
La matérialisation des pistes est fonction de la position de la tête de lecture, plus ou moins loin du centre.

Dans le **formattage Soft**, la disquette est rigoureusement vierge et inutilisable par la machine, celle-ci possède un programme utilitaire de formattage (voir le mode d'emploi de votre ordinateur) qui inscrit de façon magnétique les signaux de début de secteur.

Les deux systèmes ont des avantages et des inconvénients, le second semble devenir de plus en plus répandu.

ATTENTION ! Le formattage est une opération spécifique à une marque d'ordinateurs, elle détermine le nombre de pistes, de secteurs par pistes... paramètres qui rendent en général rigoureusement incompatibles les disquettes enregistrées sur des machines de marques différentes.

MEMOIRES

L'ordinateur a besoin de faire référence à des éléments qui seront soit des données (des valeurs), soit des programmes (la suite des opérations à effectuer). Ces éléments ne lui sont accessibles que sous forme d'impulsions électriques, elles sont stockées dans la machine, dans la mémoire centrale sous deux formes :

- ***Les mémoires mortes (ROM pour Read Only Memory).***
- ***Les mémoires vives (RAM pour Random Access Memory).***

- Les **ROM** sont en général de faible capacité. Sur les systèmes gérés sous CP/M, elles se contentent de conserver le petit

programme de départ, de démarrage de la machine. Leur grand intérêt réside dans le fait qu'elle gardent leurs informations même sans courant électrique ; en contrepartie, on ne peut y écrire - de façon définitive - qu'à leur fabrication, en usine.

Certaines machines possèdent leur langage évolué, voire leur système d'exploitation, sous cette forme, il s'agit en général de petites machines.

- Les **RAM**, en revanche, nécessitent une alimentation électrique pour conserver leurs informations. Par conséquent, on peut y écrire, y effacer, y réécrire, y lire tout ce que l'on souhaite. Lors de la coupure du courant de l'ordinateur les RAM s'effacent de manière irréversible.

Ce dernier type de mémoire est le plus intéressant, c'est lui qui déterminera la puissance de travail de la machine.

L'OCTET

On mesure la taille des mémoires en **octets**, groupe de huit bits ou informations électriques élémentaires l'octet étant l'équivalent d'un signe (lettre, chiffre ou signe). On parle couramment de machines à 8, 16 kilo octets de mémoire morte (8 ØØØ, 15 ØØØ octets) et de 16, 64 kilo octets de mémoire vive.

PISTES

La piste est la couronne circulaire, de largeur très faible qui contient les informations sous forme magnétique sur une disquette.

Contrairement à un disque microsillon qui ne possède qu'une seule piste (le sillon) en spirale, la disquette magnétique possède un certain nombre de pistes (3Ø, 32, 4Ø par exemple) qui sont des cercles concentriques.

L'accès à telle ou telle piste se fait par la translation de la tête de lecture le long d'un axe.

SECTEURS

Une piste est organisée en secteurs d'une quantité finie d'informations ; de la sorte, la machine peut accèder directement au bloc élémentaire d'informations que constitue le secteur. Par exemple, le système 12, piste 4 est parfaitement identifiable par le système d'exploitation.

TABLEAU RECAPITULATIF : ORDRE DIR

DIR␣	**(R)**	Donne tous les fichiers accessibles sur le disque actif
	N: (R)	Donne tous les fichiers accessibles sur le disque N: puis retour au disque actif
	***.BAS (R)**	Donne tous les fichiers d'extension BAS accessibles sur le disque actif
	N:*.BAS (R)	Donne tous les fichiers d'extension BAS accessibles sur le disque N: puis retour au disque actif
	FICHIER.BAS (R)	Vérifie la présence du fichier de nom FICHIER.BAS sur le disque actif
	FICHIER.* (R)	Donne tous les fichiers de nom FICHIER quelles que soient leurs extensions
	FICH???.* (R)	Donne tous les fichiers commençant par FICH, quelles que soient les extensions (ex.FICHES.BAS ou FICHIER.COM...)
	N:FICHIER.BAS (R)	Vérifie sur le disque N: l'existence du fichier FICHIER.BAS
	N:FICHIER.* (R)	Donne sur le disque N: les fichiers de nom FICHIER, quelles que soient les extensions
	N:FICH???.* (R)	Donne sur le disque B: tous les fichiers commençant par FICH, quelles que soient les extensions

(R) = touche RETURN ou ENTER

LISTER LES FICHIERS DU DISQUE ACTIF

Syntaxe

DIR (R)

```
A>DIR
A: ORDAFA       : COPIE    COM : CENTRO   COM
A: STAT     COM : MERGPRIN OVR : WSMSGS   OVR
A>_
```

Tous les fichiers accessibles au Directory (ceux qui n'ont pas été masqués par l'ordre **STAT $ R/O)** apparaissent sur une liste qui montre en début de ligne le disque étudié, et pour chaque fichier son nom et son extension.

Les fichiers sont normalement présentés par quatre de front dans la version standard du CP/M 2.2, dans les anciennes versions, on ne trouvait qu'un fichier par ligne.

Le CP/M lui-même n'apparaît pas au Directory pas plus que les ordres associés (**DIR**, **ERA**, **REN...**). Seules les commandes transitoires (STAT, PIP...) sont visibles.

DIR est donc une commande résidente.

Syntaxe

DIR␣B: (R)

```
A>DIR B:
B: DUMP     ASM : FORMGEN  COM : INSTALL  COM
B: BATCH    COM : DATASTAR COM : PIP      COM
B: CUSTOMER NDX : OKSTATES NDX : PRODUCTS NDX
B: CUSTOMER DTA : OKSTATES DTA : PRODUCTS DTA
B: ORDER        : CUSTOMER BAK : SUBMIT   COM
B: SORT     COM : STAT     COM : DUMP     COM
B: FORMAT   COM : HI1502   COM
A>_
```

Il n'y a pas de différence apparente avec l'ordre **DIR** simple, cette différence apparaît cependant au signe A>- qui apparaît avant l'ordre **DIR B:** et au même signe qui apparaît après la liste des fichiers.

Ceci indique bien que le listage du directory n'a pas modifié le numéro du lecteur actif qui reste le A.

Cet ordre charge en mémoire la liste des fichiers et l'encombrement du disque B: par la suite, par exemple, un ordre stat donnerait l'encombrement de chacun des deux disques.

LISTER DES FICHIERS D'APRES LEUR EXTENSION

Syntaxe

DIR␣*.BAS (R)

ou DIR␣N:*.BAS (R)

```
A>DIR *.BAS
A: PG55      BAS : REG80     BAS : DEMO      BAS
A: PG60      BAS : INTGR     BAS : CREF      BAS
A: LISTF     BAS : CONSF     BAS : FICHES    BAS
A: TEST      BAS
A>_
```

Tous les fichiers d'extension **BAS**, et seulement ceux-ci, apparaissent au Directory.

```
A>DIR B:*.BAS
B: JEU       BAS : PENDU     BAS : ACCUEIL   BAS
A>_
```

Tous les fichiers d'extension **BAS** existant sur le disque B: sont listés, le disque A: redevient actif ensuite.

LISTER DES FICHIERS QUELLE QUE SOIT LEUR EXTENSION

Syntaxe

DIR␣FICHIER.* (R)

ou DIR␣N:FICHIER.* (R)

```
A>DIR DUMP.*
A: DUMP      ASM : DUMP      DOC : DUMP      COM
A>_
```

Tous les fichiers de nom **DUMP** sont listés quelles que soient leurs extensions.

```
A>DIR B:PROG.*
B: PROG      ASM : PROG      BAS : PROG      COM
A>_
```

Tous les fichiers de nom **PROG** existant sur le disque B: sont listés quelles que soient leurs extensions.

Le lecteur A: reste le lecteur actif après le listage.

LISTER DES FICHIERS D'APRES LE DEBUT DE LEUR NOM

Syntaxe

DIR␣FIC????.EXT (R)
ou DIR␣N:FIC????.EXT (R)

```
A>DIR FIC?????.*
A: FICHIER      : FICHIER3 BAS : FICHE2   BAS
A: FICHIER  COM
A>_
```

Tous les fichiers dont le nom commence par **FICH** sont listés à l'écran.

Les fichiers suivants sont susceptibles d'être édités

FICHES FICHIER ...

ceci peut servir, comme dans le cas de l'extension, à sauver les versions successives de réalisation d'un programme et à les lister pour savoir à quelle version on se trouve.

```
A>DIR B:FICH????.*
B: FICHIER      : FICHIER3 BAS : FICHE2   BAS
B: FICHIER  COM
A>_
```

Tous les fichiers commençant par **FICH** existant sur le disque B: sont listés avec retour au disque A: actif.

Syntaxe

DIR␣FICHIER.EXT (R)
ou DIR␣N:FICHIER.EXT (R)

```
A>DIR MBASIC.COM
A: MBASIC    COM
A>_
```

Le fichier **MBASIC.COM** est présent sur le disque, il apparaît donc au directory.

```
A>DIR MBASIC.COM
NO FILE
A>_
```

Le fichier **MBASIC.COM** n'est pas sur le disque, le message le précise.

```
A>DIR B:MBASIC.COM
B: MBASIC    COM
A>_
```

Le Numéro d'unité (B:) placé à l'avant, précise le disque de recherche sans changer le disque actif.

LISTER LES FICHIERS PROTEGES CONTRE LE LISTAGE

CP/M 86 uniquement

Syntaxe

DIRS

Cet ordre liste les fichiers qui ont été protégés par l'ordre **STAT** de mise en système.

Exemple 1

```
A DIRS
NON-SYSTEM FILE(S) EXISTS
```

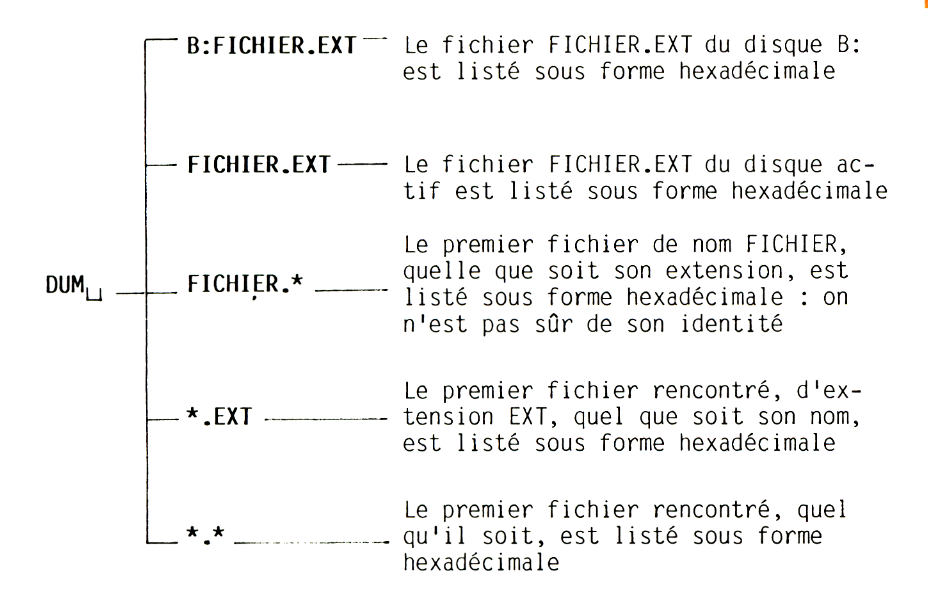
Il n'y a pas de fichiers system mais des fichiers non system existent.

Exemple 2

```
A STAT XSUB.CMD SSYS
A:XSUB      CMD set to System (Sys)
A DIRS
A:XSUB      CMD
NON-SYSTEM FILE(S) EXISTS
```

Il y a un fichier system, il y a en outre des fichiers non system.

TABLEAU RECAPITULATIF : ORDRE DUMP



Règle : le nom du fichier à lister sous forme hexadécimale doit être complet, sans ambiguïté sous peine d'obtenir un autre listage que celui que l'on souhaite.

LISTER EN HEXADECIMAL UN FICHIER

Syntaxe

DUMP␣FICHIER.EXT

```
A>DUMP PIP

NO INPUT FILE PRESENT ON DISK

A>DUMP PIP.COM

0000 C3 CE 04 C9 00 00 C9 00 00 1A 00 00 00 00 00 00
0010 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0020 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0030 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0040 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0050 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0060 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0070 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29
0080 28 49 4E 50 3A 2F 4F 55 54 3A 53 50 41 43 45 29


A>^C
A>_
```

Le fichier **PIP.COM**, pour être listé doit être décrit avec son extension, son nom seul donne une erreur.

Syntaxe

DUM␣[FICHIER].[EXT]

```
A>DUMP *.BAS

0000 FF 18 63 32 00 91 20 FF 96 28 0F 1A 29 00 3A 63
0010 64 00 82 20 49 F0 12 20 CE 20 14 3A 91 20 D0 0F

A>^C
A>_
```

Le fichier listé est le premier à extension BAS que le CP/M a trouvé.

```
A>DUMP *.*

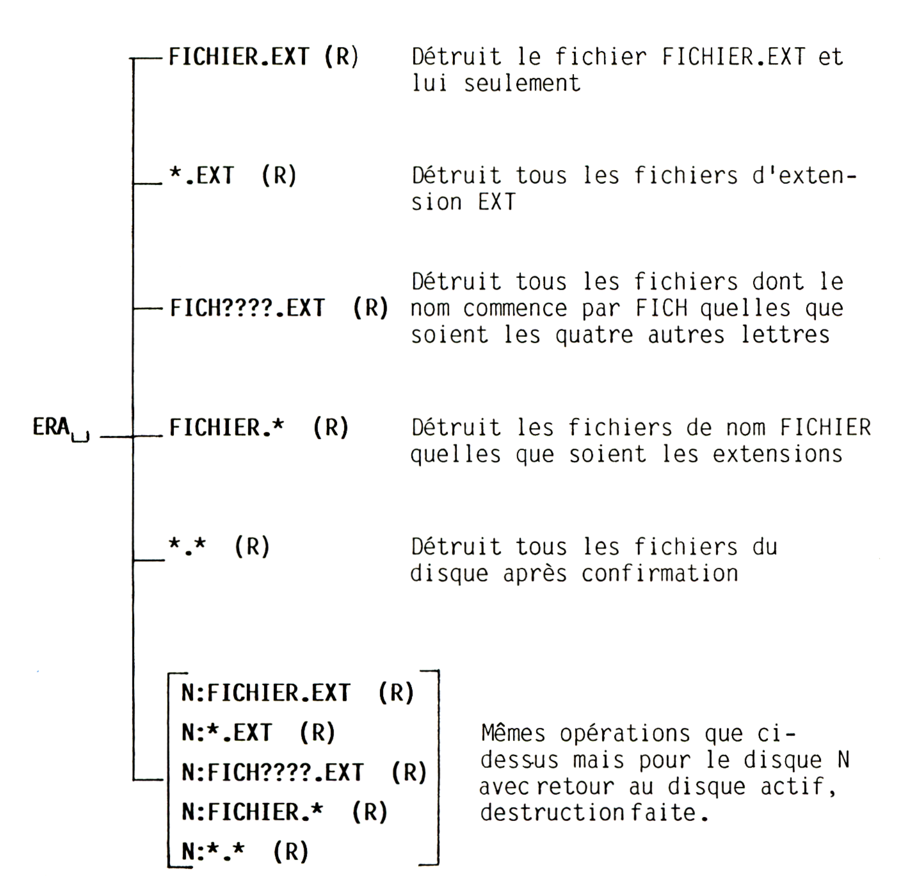
0000 3B 09 46 49 4C 45 20 44 55 4D 50 20 50 52 4F 47
0010 52 41 4D 2C 20 52 45 41 44 53 20 41 4E 20 49 4E
0020 50 55 54 20 46 49 4C 45 20 41 4E 44 20 50 52 49
0030 4E 54 53 20 49 4E 20 48 45 58 0D 0A 3B 0D 0A 3B
0040 09 43 4F 50 59 52 49 47 48 54 20 28 43 29 20 31

A>^C
A>_
```

Le fichier listé est le premier fichier trouvé par CP/M.

ATTENTION : Dans l'un ou l'autre cas, on n'est pas sûr de l'identité du fichier listé.

TABLEAU RECAPITULATIF : ORDRE ERA



Les destructions ne sont possibles que sur des disques non protégés et sur des fichiers non protégés (ordre STAT $R/O)

DETRUIRE UN FICHIER EXPLICITE

Syntaxe

ERA␣FICHIER.EXT (R)

```
A>DIR MBASIC.COM
A: MBASIC   COM
A>
A>ERA MBASIC.COM
A>
A>DIR MBASIC.COM
NO FILE
A>_
```

Le fichier **MBASIC.COM** a été supprimé.

```
A>ERA COPIE.COM
bios write error  write protected,write fault
Bdos Err On A: Bad Sector
A>_
```

La suppression n'est pas possible si la disquette est verrouillée.

```
A>STAT MBASIC.COM $R/O

MBASIC.COM set to R/O
A>ERA MBASIC.COM
Bdos Err On A: File R/O
A>_
```

Un fichier explicitement protégé n'est pas effaçable. (voir ordre **STAT $**)

DETRUIRE LES FICHIERS DE MEME EXTENSION

Syntaxe

ERA␣*.EXT (R)

```
A>DIR
A: COPIE     BAS : TEXTE     BAS : STAT      COM
A>
A>ERA *.BAS
A>
A>DIR
A: STAT      COM
A>_
```

Tous les fichiers à extension **BAS** ont été détruits.

DETRUIRE LES FICHIERS DE MEME NOM

Syntaxe

ERA␣FICHIER.* (R)

```
A>DIR
A: PROG      ASM : PROG      COM : STAT      COM
A>
A>ERA PROG.*
A>
A>DIR
A: STAT      COM
A>_
```

Tous les fichiers **PROG**, quelle que soit l'extension ont été détruits.

DETRUIRE LES FICHIERS D'APRES LE DEBUT DE LEUR NOM

Syntaxe

ERA␣FICH????.EXT (R)

```
A>DIR
A: FICHES    BAS : FICHIER  BAS : FICHE5   BAS
A: STAT      COM
A>
A>ERA FICH????.BAS
A>
A>DIR
A: STAT      COM
A>_
```

Les fichiers :

FICHES.BAS
FICHIER.BAS
FICHE5.BAS

ont été détruits.

L'ordre :

B:ERA␣FICH????.BAS

aurait produit le même effet mais sur le disque B sans changer de disque actif.

Syntaxe

ERA␣*.* (R)

```
A>ERA *.*
ALL (Y/N)?Y
A>
A>DIR
NO FILE
A>_
```

On voit que par mesure de précautions, la machine interroge par le message "ALL (Y/N) ; à la réponse Y (R), la machine détruira tous les fichiers présents sur le disque.

Si le disque possède un ou plusieurs fichiers protégés, aucun fichier ne sera détruit.

Les fichiers non apparents au Directory (ordre **STAT $SYS**) ne sont pas protégés.

Le CP/M lui-même n'est pas détruit par l'ordre de destruction totale, le disque peut donc toujours être "booté".

OBTENIR DES AIDES SUR L'UTILISATION DES ORDRES

CP/M 86 uniquement

Syntaxe

HELP nom de l'aide

Exemple

A HELP CP/M DIR

La commande DIR (suit un texte explicatif sur l'ordre) :

Aditionnal topics available

Exemples

HELP

Le texte explicatif peut induire d'autres textes : les exemples ; pour les obtenir, on pourrait frapper :

HELP CP/M DIR EXEMPLES

l'ordinateur reste en mode **HELP** pour continuer à interroger : si on tape la touche RETOUR, il repasse en CP/M 86.

Remarque : le texte explicatif est un texte écrit en général sous la responsabilité du fabricant de l'ordinateur.

Cette facilité offerte par CP/M 86 suppose pour être pleinement efficace, que le fichier de textes ait été introduit par le fabricant de la machine.

TABLEAU RECAPITULATIF : ORDRE N:

N: Rend le lecteur de disque numéro N actif.

Tous les ordres qui seront passés sous CP/M sans préciser le numéro du lecteur seront effectués sur le lecteur N.

On dit que N est le disque actif (on dit parfois que N est le disque loggé).

CHANGER LE DISQUE ACTIF

```
A>B:
B>DIR
B: DEBUT      BAS : DEPART    BAS : DEBUG     COM
B: STAT       COM : ED        COM
B>_
```

B devient actif, les ordres à unité implicite sont traités sur le disque B.

Si l'on veut effectuer une opération sur un fichier qui se trouve sur A, on doit le préciser par exemple **A:MBASIC** pour charger le Basic.

TABLEAU RECAPITULATIF : ORDRE PIP

Syntaxe

- PIP N:=M:FICHIER.EXT		Copie d'un disque M au disque N sans changer de nom
- PIP N:FIC1.EXT=FIC2.EXT		Copie d'un disque actif au disque N en changeant le nom
- PIP UNITE:=FICHIER.EXT		Passe sur l'unité UNITE le fichier FICHIER.EXT
- PIP FICHIER.EXT=UNITE:		Crée le fichier FICHIER.EXT à partir de l'unité UNITE
- PIP UNITE:=UNITE:		Les manifestations de l'unité de gauche passent sur l'unité de droite
- PIP FICHIER.EXT=FICHIER.EXT	|V|	Copie d'un fichier avec vérification
	|Dx|	Copie avec destruction de tout ce qui est après la Xième colonne
	|E|	Copie avec affichage témoin à l'écran
	|L|	Copie avec conversion en minuscules
	|Gx|	Copie d'après la zone utilisateur X
	|N|	Copie en ajoutant les numéros de lignes
	|Qchaine Z|	Copie jusqu'à la suite de lettres chaîne comprise
	|Schaine Z|	Copie à partir de la suite de lettres chaîne comprise

TABLEAU RECAPITULATIF : ORDRE PIP

- PIP FICHIER.EXT=FICHIER.EXT	|U|	Copie en changeant les minuscules en majuscules
	|F|	Copie en supprimant les caractères de saut de page
	|P|	Copie en insérant un saut de page toutes les x lignes
	|H|	Copie un fichier hexadécimal en vérifiant si les codes sont au format INTEL
	|Tx|	Copie en créant une tabulation toutes les x colonnes
	|W|	Copie en ignorant les attributs $R/O des fichiers

Les unités	CON:	La console (l'écran clavier)
	LST:	L'imprimante
	RDR:	Lecteur de ruban (voir votre machine)
	PUN:	Perforateur de ruban (voir votre machine)

Syntaxe

PIP␣A:=B:PROG.EXT
PIP␣A:PROGNOU.EXT=B:PROGANC.EXT

```
A>DIR
A: DUMP      ASM : PIP       COM : CENTRO  COM
A>DIR B:
B: ED        COM
A>PIP B:=A:CENTRO.COM

A>DIR
A: DUMP      ASM : PIP       COM : CENTRO  COM
A>DIR B:
B: CENTRO    COM : ED        COM
A>_
```

L'ordre **PIP B:=A:CENTRO.COM** a copié le fichier sur le disque B, sous le même nom.

Le catalogue du disque B après l'opération confirme le transfert.

Avec changement de nom :

```
A>DIR
A: DUMP      ASM : PIP       COM : CENTRO    COM
A>DIR B:
B: ED        COM
A>PIP B:IMPR.COM=A:CENTRO.COM

A>DIR
A: DUMP      ASM : PIP       COM : CENTRO    COM
A>DIR B:
B: IMPR      COM : ED        COM
A>_
```

Le fichier **CENTRO.COM** du disque A a été copié sur le disque B sous le nom **IMPR.COM**.

COPIER DES FICHIERS D'UN DISQUE SUR L'AUTRE D'APRES LEUR EXTENSION

Syntaxe

PIP␣A:=B:*.EXT

```
A>DIR
A: DUMP      ASM : PIP      COM : CENTRO   COM
A>DIR B:
B: ED        COM
A>PIP B:=A:*.COM

COPYING -
PIP.COM
CENTRO.COM

A>DIR
A: DUMP      ASM : PIP      COM : CENTRO   COM
A>DIR B:
B: PIP       COM : CENTRO   COM : ED       COM
A>_
```

Tous les fichiers d'extension COM ont été copiés du disque A sur le disque B.

Les noms des fichiers n'ont pas été changés.

La machine, au cours de l'opération de copie, nomme les fichiers qu'elle transfère.

Cette facilité est intéressante pour copier un type de fichiers sur lequel on a travaillé.

Syntaxe

PIP␣A:=B:*.*

```
A>DIR
A: DUMP     ASM : PIP      COM : CENTRO   COM
A>DIR B:
B: CENTRO   COM : ED       COM
A>PIP B:=A:*.*

COPYING -
DUMP.ASM
PIP.COM
CENTRO.COM

A>DIR B:
B: DUMP     ASM : PIP      COM : CENTRO   COM
 : ED       COM
A>_
```


Tous les fichiers du disque A ont été transférés sur le disque B.

A la différence de la copie piste à piste, cet ordre réorganise l'espace disque, le disque B est donc soit la somme des fichiers qu'il contenait plus les fichiers de l'autre disque, soit, dans le cas où il ne contenait rien (mais était formatté), tous les fichiers du disque A, réorganisés : les directory sont donc apparemment semblables mais l'occupation du disque peut être différente.

Durant le travail, l'ordre **PIP** complet explicite son travail en nommant les fichiers qu'il est en train de transférer.

Le transfert ne peut se faire que sur une disquette formattée mais la présence de CP/M n'est pas obligatoire.

Le fichier CENTRO.COM qui existait déjà a été remplacé par sa copie conforme.

PLUSIEURS OPERATIONS PIP SUCCESSIVES

Syntaxe

PIP

```
A>DIR
A: DUMP        ASM : CENTRO    COM
A>DIR B:
B: PIP         COM : CENTRO    COM : ED        COM
A>B:PIP
*EDIT.COM=B:ED.COM
*^C
A>DIR
A: DUMP        ASM : EDIT      COM : CENTRO    COM
A>_
```

L'ordre **B:PIP** appelle la commande **PIP** qui se trouve sur le disque B ; si cette commande avait été sur le disque A, le simple ordre PIP aurait suffi.

On vérifie que sur les disques A et B des fichiers existent.

Une étoile apparaît en début de ligne indiquant que la machine attend un ordre **PIP**.

L'ordre **EDIT.COM=B:ED.COM** va transférer sur le disque actif (le A) le fichier ED.COM qui se trouve sur le B (début de fichier B:) en l'appelant EDIT.COM.

Une nouvelle étoile apparaît en début de ligne indiquant la possibilité d'introduire une nouvelle commande PIP.

Un directory du disque actif laisse apparaîtreun nouveau fichier EDIT.COM qui est le programme ED.COM du disque B rebaptisé.

L'ordre C rend la main à CP/M.

Cette procédure de transferts en cascade est très utile dans le cas de copie de certains fichiers d'un disque à l'autre sans avoir à relancer la procédure **PIP** qui est active tout le temps du transfert.

Syntaxe

PIP␣LST:=CON:

```
ON ECRIT DIRECTEMENT SUR L'IMPRIMANTE
```

```
A>PIP LST:=CON:
ON ECRIT DIRECTEMENT SUR L'IMPRIMANTE
A>_
```

La première ligne donne le texte tel qu'il apparaît sur l'imprimante.

Les lignes suivantes donnent la recopie d'écran des séquences d'ordres qui ont amené à ce résultat. L'ordre :

PIP LST:=CON:

signifie que l'unité **LST:** c'est-à-dire l'unité de listage (l'imprimante) est destinataire de l'unité **CON:**, c'est-à-dire le clavier. Tout caractère frappé sur le clavier est donc immédiatement écrit sur l'imprimante.

Tous les caractères doivent être tapés, dans l'exemple, le mot "IMPRIMANTE" doit être suivi des caractères "retour chariot" ou "return"... puis du caractère "Line feed" pour obtenir le résultat affiché. Ces caractères indispensables n'apparaissent pas comme tel dans le texte mais évitent pour l'un de dépasser la largeur définie et pour l'autre de réécrire sur le texte déjà écrit.

COPIER UN FICHIER AVEC ECHO A L'ECRAN DE CE QUI EST COPIE

Syntaxe

PIP␣A:=B:FICH.EXT |E|

```
A>DIR
A: PIP      COM
A>DIR B:
B: TEXTE    TEX : CENTRO   COM
A>PIP A:=B:TEXTE.TEX[E]
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>
```

```
A>DIR
A: PIP      COM : TEXTE    TEX
A>
```

L'imprimante est mise en double sortie de l'écran (ordre contrôle P de CP/M).

Le disque A ne possède que le fichier PIP.COM.

Le disque B possède les fichiers TEXTE.TEX et CENTRO.COM.

L'ordre **PIP** de copie du fichier TEXTE.TEX du disque B sur le disque A est suivi de l'extension PIP E (pour Echo).

Durant la copie, tout caractère transféré est en même temps écrit s'il est affichable.

On vérifie que le fichier TEXTE.TEX est maintenant sur le disque A.

ATTENTION à la syntaxe, l'extension PIP doit être entre crochets.

Syntaxe

PIP␣A:=B:FICH.EXT |V|

```
A>PIP A:=B:COURBE.BAS

A>ERA COURBE.BAS
A>
A>PIP A:=B:COURBE.BAS[V]

A>_
```

Le paramètre V mis entre crochets à la fin de l'ordre **PIP** a pour effet de forcer une vérification de la correspondance du fichier copié avec le fichier à copier (le fichier source).

Dans notre exemple, le premier ordre **PIP** s'est effectué en 15 secondes 9 dixièmes, il s'agit d'une copie simple.

Le second ordre **PIP** suivi du paramètre de vérification V s'est accompli en 2Ø secondes et 3 dixièmes, le temps de la vérification s'est ajouté.

La vérification accomplie se fait sans contrôle à l'écran, par contre, si la copie s'avère mauvaise, un message en avertit l'utilisateur à l'écran.

L'ordre **PIP** paramétré en V est à utiliser systématiquement lors de copie de fichiers longs et importants, en règle générale, la fiabilité des lecteurs est suffisante pour pouvoir s'en passer dans la plupart des cas, surtout pour les fichiers courts.

RASSEMBLER PLUSIEURS FICHIERS SOUS UN SEUL NOM

Syntaxe

PIP␣A:FICHIER.EXT=A:FIC1.EXT.A:FIC2.EXT ...

```
A>DIR
A: PIP        COM
A>PIP TEXT1.TEX=CON:
CECI EST UNE PREMIERE LIGNE DE TEXTE
A>PIP TEXT2.TEX=CON:
CECI EST UNE SECONDE LIGNE
A>PIP B:TEXTE.TEX=A:TEXT1.TEX,A:TEXT2.TEX

A>DIR B:
B: PIP        COM : TEXTE    TEX : CENTRO   COM
A>TYPE B:TEXTE.TEX
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>_
```

Le disque A ne possède initialement qu'un seul fichier : PIP.COM.

On crée deux fichiers TEXT1 et TEXT2 grâce à l'ordre PIP de création de fichier depuis la console (PIP NOM=CON:).

On associe ces deux fichiers sous le nom TEXTE.TEX avec l'ordre PIP ; à gauche du signe égal se trouve le nom du fichier à créer, à droite se trouve le nom des fichiers à regrouper, dans l'ordre où ils seront regroupés et séparés par une virgule.

L'ordre **DIR** du disque B (le nom du fichier créé commençant par A:, il a été sauvé sur le disque B) laisse apparaître un fichier TEXTE.TEX.

La sortie à l'écran (ordre TYPE) montre que ce nouveau fichier est la juxtaposition des fichiers TEXT1 et TEXT2.

Cette procédure n'est active que pour des fichiers en format ASCII.

Syntaxe

PIP␣CON:=A:FICH.EXT

```
A>DIR
A: DUMP      ASM : PIP      COM : CENTRO   COM
A>PIP CON:=DUMP.ASM
;       FILE DUMP PROGRAM, READS AN INPUT FILE
;
;       COPYRIGHT (C) 1975, 1976, 1977, 1978
;       DIGITAL RESEARCH
;       BOX 579, PACIFIC GROVE
;       CALIFORNIA, 93950
;
        ORG     100H
BDOS    EQU     0005H   ;DOS ENTRY POINT
CONS    EQU     1       ;READ CONSOLE
ABORTED: DUMP.ASM

A>_
```


PIP étant un ordre de transfert d'une unité vers une autre, il suffit de désigner la première unité (celle qui reçoit) comme étant la console (CON:) et la seconde comme un fichier (ici le fichier DUMP.ASM sur le disque actif).

Le fichier, comme dans l'ordre TYPE, ne peut être listé que sous ASCII, tout fichier sauvé sous forme abrégée apparaîtra comme incompréhensible.

Dans l'exemple, la mention ABORTED: signale que l'opération a été arrêtée accidentellement en cours d'exécution, en fait, pour limiter l'exemple, la touche BREAK (ou comtrol C) a été enfoncée en cours de listage pour ne pas avoir un listage trop long.

TRANSFORMER TOUTES LES LETTRES D'UN FICHIER EN MINUSCULES OU EN MAJUSCULES

Syntaxe

PIP␣A:FIC1.EXT=A:FIC.EXT|U|

ou PIP␣A:FIC1.EXT=A:FIC.EXT|L|

```
A>DIR
A: PIP      COM : TEXTE    TEX
A>
A>TYPE TEXTE.TEX .
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>
A>PIP TEXTEBIS.TEX=TEXTE.TEX[L]

A>TYPE TEXTEBIS.TEX
ceci est une premiere ligne de texte
ceci est une seconde ligne
A>_

A>TYPE TEXTE.TEX
ceci est une premiere ligne de texte
ceci est une seconde ligne
A>
A>PIP TEXTEMAJ.TEX=TEXTE.TEX[U]

A>TYPE TEXTEMAJ.TEX
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>_
```

Dans la première partie, on transforme **TEXTE.TEX** en **TEXTEBIS.TEX** qui est mis en minuscules.

Dans la seconde partie, le fichier **TEXTEBIS.TEX** qui a été entre temps renommé en **TEXTE.TEX** est à nouveau transformé en majuscules.

Cet ordre n'est évidemment possible qu'avec des signes ASCII donc uniquement sur les textes ou programmes sauvés en ASCII.

Syntaxe

PIP␣A:FIC1.EXT=A:FIC.EXT|N|

```
A>DIR
A: PIP      COM : TEXTE    TEX
A>
A>TYPE TEXTE.TEX
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>
A>PIP TEXTE1.TEX=TEXTE.TEX[N]

A>TYPE TEXTE1.TEX
     1: CECI EST UNE PREMIERE LIGNE DE TEXTE
     2: CECI EST UNE SECONDE LIGNE
A>_
```

L'ordre **PIP** permet de numéroter les lignes d'un fichier, le séparateur de lignes est le caractères Retour Chariot (Return, CR ou Enter...).

Le fichier TEXTE.TEX est un fichier normal, créé antérieurement.

L'ordre **PIP** TEXTE1.TEX=TEXTE.TEX|N| crée un fichier TEXTE1.TEX dont chaque ligne est numérotée.

La numérotation se fait selon le standard CP/M, chiffre des unités sur la 6e colonne puis deux points, puis un espace puis la ligne proprement dite.

Cette numérotation prenant donc de la place supplémentaire est à manier avec précautions, elle peut cependant rendre d'appréciables services.

Si au lieu de N on avait utilisé 2N, on aurait eu les zéros à gauche du chiffre :

```
A>PIP TEXTE1.TEX=TEXTE.TEX[N2]

A>TYPE TEXTE1.TEX
000001  CECI EST UNE PREMIERE LIGNE DE TEXTE
000002  CECI EST UNE SECONDE LIGNE
```

PIP

COPIER LES DEBUTS DE LIGNES D'UN FICHIER

Syntaxe

PIP␣A:FIC1.EXT=A:FIC.EXT|DNN|

où **NN** est le nombre de caractères par ligne

```
A>DIR
A: PIP        COM
A>
A>DIR B:
B: TEXTE      TEX : CENTRO    COM
A>
A>PIP TEXTE.TEX=B:TEXTE.TEX[D20]

A>TYPE TEXTE.TEX
CECI EST UNE PREMIER
CECI EST UNE SECONDE
A>_
```

Le directory du disque A laisse apparaître un seul fichier PIP.COM.

Le directory de B laisse apparaître deux fichiers TEXTE.TEX et CENTRO.COM.

Le fichier TEXTE.TEX est celui qui a été créé par l'ordre de fusion de plusieurs fichiers (voir "Rassembler plusieurs fichiers sous un seul nom").

On copie ensuite à l'aide de PIP le fichier TEXTE.TEX du disque B sur le disque actif (le A) avec l'extension PIP D2Ø qui demande à PIP de ne copier que les 2Ø premiers signes du fichier origine. La valeur 2Ø est prise à titre d'exemple, n'importe quelle valeur serait correcte.

L'ordre de sortie à l'écran TYPE montre le résultat.

Syntaxe

PIP␣PRN:=CON:

```
     1: AAAAAAAAAAAAAAAAAAAAA  AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA  AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
     2: BBBBBBBBBBBBBBBBBBB  BBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBB



A>PIP PRN:=CON:
AAAAAAAAAAAAAAAAAAAAAAAAAAAA  AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA  AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  BBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBB

A>_
```

Le premier bloc de texte montre le résultat sur l'imprimante, on remarque que chaque ligne commence par son numéro d'ordre.

Chaque nouvelle ligne commence après le caractère Retour Chariot (Return ou Enter).

Le caractère Line Feed de passage à la ligne suivante d'écran n'est pas interprété comme la fin de la ligne à numéroter.

La seule différence entre l'ordre **PIP LST:=CON:** et l'ordre **PIP PRN:=CON:** est donc que le second organise les lignes en blocs numérotés alors que le premier se contente de reproduire à l'imprimante ce qui sort à l'écran.

Le second bloc de texte est la recopie d'écran correspondant au résultat du premier bloc. On voit que l'imprimante n'est pas la copie de l'écran.

CREER UN FICHIER DE TEXTE AVEC PIP

Syntaxe

PIP␣A:FIC.EXT=CON:

```
DIR
A: PIP        COM : CENTRO    COM


A>PIP ENTREE.TEX=CON:
Grace à CP/M et à la commande transitoire PIP,
il est possible de faire tous les échanges
entre les périphériques connus du système.


A>DIR
A: PIP        COM : CENTRO    COM : ENTREE    TEX
A>TYPE ENTREE.TEX
Grace à CP/M et à la commande transitoire PIP,
il est possible de faire tous les échanges
entre les périphériques connus du système.

A>_
```

L'ordre **PIP** permet de créer des fichiers de texte.

Il est certes plus commode, lorsque l'on s'en sert avec assurance, d'utiliser l'éditeur de texte ED. La simplicité de mise en oeuvre de PIP en fait cependant un outil intéressant pour créer des fichiers explicatifs, des aide-mémoire en quelques sortes. On peut par exemple créer sous cette forme un fichier FICHIER.TEX qui est le mode d'emploi du programme FICHIER.BAS.

Dans l'exemple ci-dessus, on définit un fichier ENTREE.TEX comme l'unité (ici le fichier) destinataire et l'unité CON: (donc le clavier) comme la source.

Le fichier ENTREE.TEX ainsi créé contient les mots qui ont été frappés au clavier comme le montre l'ordre TYPE qui le liste à l'écran.

La frappe doit contenir tous les codes, par exemple une fin de ligne doit se noter par les deux caractères Retour chariot (RETURN ou ENTER) puis Passage de Ligne (Line FEED ou LF).

Syntaxe

PIP␣FIC1.EXT=FIC.EXT|QCHAINE ↑ Z|

```
A>DIR
A: PIP        COM : TEXTE    TEX
A>TYPE TEXTE.TEX
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>
A>PIP TEXTE1.TEX=TEXTE.TEX[QSECON^Z]

A>TYPE TEXTE1.TEX
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECON

A>_
```

PIP

La copie se fait en fait jusqu'à un groupe de lettres qui peut être ou non un mot.

Dès que la machine rencontre ce mot, elle le transfère puis stoppe la copie.

L'ordre :

PIP TEXTE1.TEX = TEXTE.TEX|QSECON|Z|

signifie Quitte à SECON ; Le signe Controle Z signale la fin du groupe de lettres.

Le fichier **TEXTE1.TEX** est la copie de **TEXTE.TEX** mais arrêté à SECON inclus.

COPIER UN FICHIER A PARTIR D'UN MOT

Syntaxe

PIP␣FIC1.EXT=FIC.EXT[SCHAINE↑Z]

```
A>DIR
A: PIP      COM : TEXTE    TEX
A>
A>TYPE TEXTE.TEX
CECI EST UNE PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>
A>PIP TEXTE1.TEX=TEXTE.TEX[SPREM^Z]

A>TYPE TEXTE1.TEX
PREMIERE LIGNE DE TEXTE
CECI EST UNE SECONDE LIGNE
A>_
```

Le point de départ de la copie peut être un mot ou n'importe quel assemblage de lettres.

Le début de la copie se fait à partir du mot compris.

Le fichier TEXTE.TEX sert de point de départ.

On copie grâce à **PIP** ce fichier sur le fichier TEXTE1.TEX avec l'extension de fin de ligne :

|SPREM|Z|

qui signifie Start PREM ; le signe Controle Z est indispensable pour signaler la fin de la chaîne de lettres de référence.

Le fichier TEXTE1.TEX, visualisé après, montre qu'il a effectivement été copié à partir des lettres PREM.

Syntaxe

PIP␣FIC1.EXT=FIC.EXT[SCHAINE↑ZQCHAINE↑Z]

```
A>DIR
A: PIP      COM : ED      COM
A>DIR B:
B: ESSAI
A>TYPE B:ESSAI
VOICI UN TEXTE D'ESSAI, IL A ETE FRAPPE SOUS
L'EDITEUR ED ET VA NOUS SERVIR A VERIFIER UN
CERTAIN NOMBRE DE POSSIBILITéS DE L'ORDRE PIP.
NOUS VERRONS NOTAMMENT COMMENT ON PEUT GROUPER
LES PARAMETRES ENTRE CROCHETS.

A>PIP A:=B:ESSAI[SL'EDIT^Z QIP.^Z]

A>TYPE ESSAI
L'EDITEUR ED ET VA NOUS SERVIR A VERIFIER UN
CERTAIN NOMBRE DE POSSIBILITéS DE L'ORDRE PIP.

A>_
```

Le groupement du paramètre de débur de copie (S) et de celui de fin de copie (Q) permet de copier une partie seulement de fichier.

Dans notre exemple, nous copions depuis la chaîne L'EDIT (donc depuis le mot L'EDITEUR) jusqu'à la chaîne IP. (donc le mot PIP.)

On voit que les chaînes sont indépendantes des mots, elles peuvent contenir une partie d'un mot ou au contraire plusieurs mots.

COMBINAISON DE PARAMETRES PIP

Syntaxe

PIP␣A:FIC.EXT=B:FIC1.EXT[ABC...]

où A,B,C... sont des paramètres de PIP

```
A>DIR
A: PIP      COM : ED       COM
A>DIR B:
B: ESSAI
A>TYPE B:ESSAI
VOICI UN TEXTE D'ESSAI, IL A ETE FRAPPE SOUS
L'EDITEUR ED ET VA NOUS SERVIR A VERIFIER UN
CERTAIN NOMBRE DE POSSIBILITés DE L'ORDRE PIP.
NOUS VERRONS NOTAMMENT COMMENT ON PEUT GROUPER
LES PARAMETRES ENTRE CROCHETS.

A>PIP A:=B:ESSAI[LN]

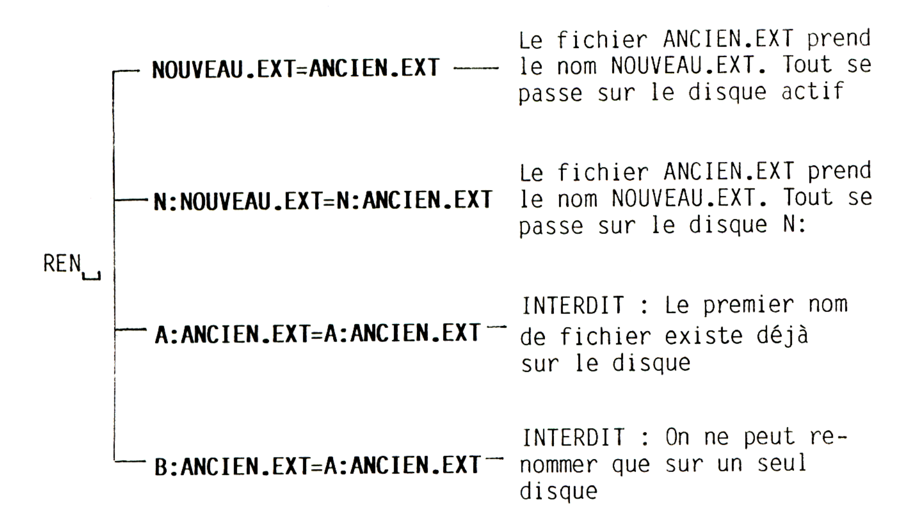
A>TYPE ESSAI
      1: voici un texte d'essai, il a ete frappe sous
      2: l'editeur ed et va nous servir a verifier un
      3: certain nombre de possibilités de l'ordre pip.
      4: nous verrons notamment comment on peut grouper
      5: les parametres entre crochets.
      6:
A>_
```

Les paramètres de **PIP** peuvent être groupés entre les deux crochets, il y a combinaison des effets des paramètres.

Dans notre exemple, les deux paramètres L et N sont groupés, l'effet est donc, comme on peut le vérifier dans le second ordre de sortie du texte (TYPE), d'abord de transformer les majuscules en minuscules et (paramètre L) puis de numéroter les lignes (paramètre N).

Tous les ordres sont en général composables, il faut cependant manier cette possibilité de **PIP** de grouper les paramètres avec une certaine prudence.

TABLEAU RECAPITULATIF : ORDRE REN



Règles :
- Le nouveau nom doit être en premier.
- Le nouveau et l'ancien nom doivent être référencés sur le même disque (voir l'ordre PIP pour le transfert).
- Si le changement de nom se fait sur le disque actif, le nom du disque peut être omis.
- Le nouveau nom de fichier ne doit pas déjà exister au directory du disque.

RENOMMER UN FICHIER SUR LE DISQUE ACTIF

Syntaxe

REN␣NOUVEAU.EXT=ANCIEN.EXT

```
A>DIR
A: DUMP       ASM : REG80      BAS : DEMO       BAS
A: CREF       BAS : CONSF      BAS : RENT       BAS
A: HORL       DOC
A>
A>REN TIME.DOC=HORL.DOC
A>
A>DIR
A: DUMP       ASM : REG80      BAS : DEMO       BAS
A: CREF       BAS : CONSF      BAS : RENT       BAS
A: TIME       DOC
A>_
```

Le fichier **TIME.DOC** s'est substitué au fichier **HORL.DOC.**

Le Directory ne laisse plus apparaître trace du titre HORL.DOC.

```
A>DIR
A: DUMP       ASM : REG80      BAS : DEMO       BAS
A: CREF       BAS : CONSF      BAS : RENT       BAS
A: TIME       DOC
A>
A>REN DEMO.BAS=CREF.BAS
FILE EXISTS
A>_
```

Le fichier **DEMO.BAS** existe déjà, la machine refuse donc de donner ce nom à un nouveau fichier.

Syntaxe

REN␣N:NOUVEAU.EXT=N:ANCIEN.EXT

```
A>DIR B:
B: XSUB       COM : CHARGE    COM : DDT        COM
A>
A>REN B:LOAD.COM=CHARGE.COM
A>
A>DIR B:
B: XSUB       COM : LOAD      COM : DDT        COM
A>_
```

Le disque A est actif (A>-), on renomme sur le disque B le fichier CHARGE.COM qui devient LOAD.COM comme il apparaît sur le second Directory.

```
A>DIR
A: DUMP       ASM : REG80     BAS : DEMO       BAS
A: CREF       BAS : CONSF     BAS : RENT       BAS
A: TIME       DOC
A>
A>DIR B:
B: XSUB       COM : LOAD      COM : DDT        COM
A>
A>REN B:HORL.DOC=A:HORL.DOC
A:HORL.DOC?

A>_
```

Une tentative de renommer sur le disque B un fichier existant sur le disque A se solde par une impossibilité.

Le transfert d'un disque sur l'autre avec changement de nom ne peut se faire qu'avec l'ordre **PIP**.

REN

RENOMMER UN FICHIER PROTEGE

Syntaxe

REN␣NOUVEAU.EXT=ANCIEN.EXT

```
A>STAT DDT.COM $R/O

DDT.COM set to R/O
A>
A>REN DEBUG.COM=DDT.COM
Bdos Err On A: File R/O
A>_
```

Un fichier protégé ne peut pas être renommé.

```
A>STAT *.*

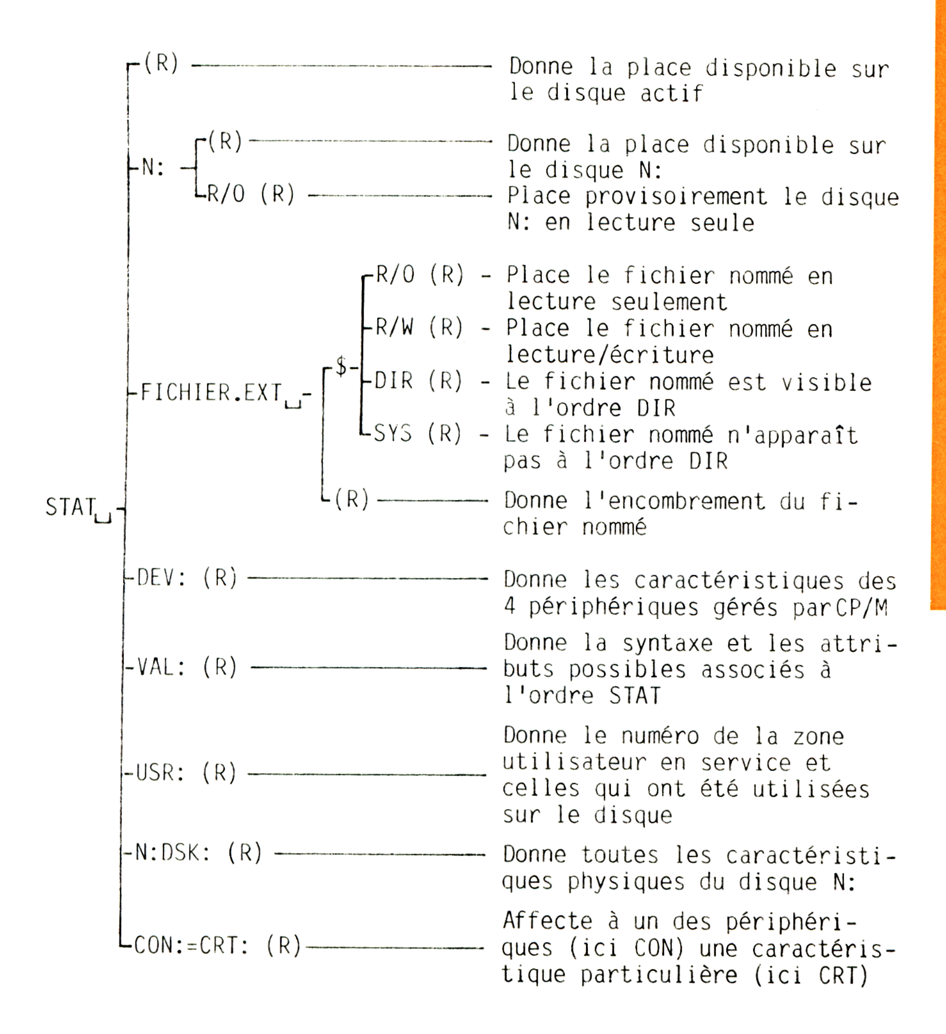
 Recs  Bytes  Ext Acc
   38     5k    1 R/O A:DDT.COM
   33     5k    1 R/W A:DUMP.COM
   41     6k    1 R/W A:STAT.COM
Bytes Remaining On A: 220k

A>STAT DDT.COM $R/W

DDT.COM set to R/W
A>
A>REN DEBUG.COM=DDT.COM
A>
A>DIR
A: DEBUG    COM : DUMP     COM : STAT     COM
A>_
```

L'ordre **STAT** général (*.*) faisant apparaître un fichier protégé, pour le renommer, il faut d'abord le repasser en non protégé (STAT fichier.ext $R/W).

TABLEAU RECAPITULATIF : ORDRE STAT



OBTENIR LA PLACE DISPONIBLE D'UN DISQUE

Syntaxe

STAT
STAT␣D:

```
A>STAT
A: R/W, Space: 160k

A>_
```

L'ordre **STAT** simple donne pour le disque actif :

- sa situation en écriture/lecture (R/W) ou en lecture seule (R/O).
- la place en kilo octets restant disponible.

```
A>B:
B>STAT
A: R/W, Space: 160k
B: R/W, Space: 151k

B>_
```

Si le disque A est actif et si on rend le disque B actif à son tour, l'ordre simple STAT donne les caractéristiques des deux disques.

Un ordre **STAT** sous A actif donnera le même résultat si le B a été rendu actif antérieurement (ordre DIR B: par exemple).

```
A>STAT A:

Bytes Remaining On A: 160k

A>_
```

L'ordre **STAT** suivi de l'indicatif du lecteur (ici A:) donne une réponse différente : seule la place disponible apparaît.

Syntaxe

STAT␣D:DSK:

```
A>STAT A:DSK:

     A: Drive Characteristics
  1904: 128 Byte Record Capacity
   238: Kilobyte Drive  Capacity
    64: 32  Byte Directory Entries
    64: Checked  Directory Entries
   128: Records/ Extent
     8: Records/ Block
    56: Sectors/ Track
     1: Reserved Tracks

A>_
```

A: est le nom du disque analysé.

1904: est le nombre d'unités élémentaires de 128 octets disponibles sur le disque (c'est les Recs de la réponse à l'ordre STAT nom de fichier vu plus loin.

238: C'est le nombre de kilo octets accessibles sur le disque (disponibles ou non).

64: C'est le nombre d'entrées directory, existantes et vérifiées, c'est donc aussi le nombre maximum de fichiers que l'on pourra stocker sur le disque.

128: Est le nombre d'enregistrements adressables par une seule entrée Directory.

8: Nombre d'enregistrements par secteurs, ici 8 x 128 soit 1Ø24 octets par secteur.

56: Est le nombre de secteurs par piste, on a donc 57344 octets par piste soit 56 kilo octets (un kilooctet vaut 1Ø24 octets).

1: Est le nombre de pistes réservées pour le système.

OBTENIR LES CARACTERISTIQUES PHYSIQUES D'UN FICHIER

Syntaxe

STAT␣FICHIER.EXT

```
A>STAT A:MBASIC.COM

 Recs  Bytes  Ext Acc
  194    25k    2 R/W A:MBASIC.COM
Bytes Remaining On A: 160k

A>_
```

Pour le fichier **MBASIC.COM**, on obtient les informations suivantes :

Recs: 194 Le fichier occupe 194 secteurs élémentaires comportant chacun 128 octets.

Bytes: 25K Le fichier occupe 25 kilo octets.

Ext: 2 Il utilise 2 blocs de gestion de 25 kilo octets, ce qui ne veut pas dire qu'il utilise 5Ø kilo octets sur le disque, ces blocs n'occupant qu'une place fictive dans le système d'exploitation.

Acc: R/W Le fichier est en lecture/écriture.

Le système après avoir rappelé le nom du fichier **A:MBASIC.COM** indique pour mémoire la place disponible sur le disque.

Si le fichier avait été rendu opaque au directory (ordre **STAT $SYS**, son nom apparaitrait entre parenthèses.

Syntaxe

STAT␣D:*.*

```
A>STAT A:*.*

 Recs   Bytes   Ext Acc
   64      8k     1 R/W A:ASM.COM
   12      2k     1 R/W A:COPYDISK.COM
   12      2k     1 R/W A:COPYSYST.COM
   38      5k     1 R/W A:DDT.COM
   33      5k     1 R/W A:DUMP.ASM
    4      1k     1 R/W A:DUMP.COM
   52      7k     1 R/W A:ED.COM
   14      2k     1 R/W A:FORMAT.COM
   14      2k     1 R/W A:LOAD.COM
  194     25k     2 R/W A:MBASIC.COM
   58      8k     1 R/W A:PIP.COM
   41      6k     1 R/W A:STAT.COM
   10      2k     1 R/W A:SUBMIT.COM
    6      1k     1 R/W A:XSUB.COM
Bytes Remaining On A: 160k

A>_
```

Tous les fichiers du disque ont été analysés, c'est le document type que l'on classe dans la pochette du disque, il permet à tout moment, sans avoir à charger le disque, de connaître en détail son contenu.

STAT

OBTENIR LES POSSIBILITES DE L'ORDRE

Syntaxe

STAT␣VAL:

```
A>STAT VAL:

Temp R/O Disk: d:=R/O
Set Indicator: d:filename.typ $R/O $R/W $SYS $DIR
Disk Status  : DSK: d:DSK:
User Status  : USR:
Iobyte Assign:
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
A>_
```

1ère ligne : Possibilité d'écriture sur le disque sous la forme d:=R/O où d est la lettre du lecteur (A ou B en général).

2e ligne : Possibilités d'affectation de paramètres aux fichiers :
Le nom du fichier suivi des quatre paramètres possibles : $R/O, $R/W, $SYS, $DIR.

3e ligne : Syntaxes possibles pour obtenir l'état du disque actif ou d'un autre disque.

4e ligne : Syntaxe pour l'obtention des zones utilisateur.

Lignes suivantes : Elles donnent les syntaxes possibles pour l'affectation de périphériques sur les quatre sorties logiques.

L'ordre **STAT VAL:** est donc un aide mémoire des syntaxes de l'ordre **STAT** en général.

Syntaxe

STAT␣USR:

```
A>STAT USR:

Active User : 0
Active Files: 0 5 9
A>_
```

Comme nous le verrons dans l'ordre **USER de CP/M**, le disque peut avoir plusieurs zones utilisateurs dans lesquelles l'opérateur peut aller travailler.

Ici, nous voyons (première ligne) que la zone utilisateur actuellement active est la zone zéro (celle qui est automatiquement active par défaut à la mise sous tension).

Nous voyons aussi que trois zones possèdent des fichiers (donc ont été effectivement utilisées), il s'agit des zones Ø, 5 et 9.

CONNAITRE LES CARACTERISTIQUES PHYSIQUES DES PERIPHERIQUES

Syntaxe

STAT␣DEV:

```
A>STAT DEV:
CON: is TTY:
RDR: is TTY:
PUN: is TTY:
LST: is TTY:

A>_
```

Les quatre périphériques ont l'affectation standard que l'on retrouve à la mise sous tension de la plupart des machines.

Pour les quatre sorties logiques, les codes sont les suivants :

CON : pour console (terminal écran/clavier).

RDR : pour lecteur de rubans perforés.

PUN : pour perforateur de rubans.

LST : sortie de listes (imprimante).

Ces termes font partie de l'histoire de l'informatique, il faut les considérer ici comme des aide-mémoire caractérisant les quatre sorties.

Pour les périphériques, les codes sont les suivants :

TTY : télétype, faible vitesse.

CTR : terminal écran/clavier.

BAT : lecteur par blocs (rubans ou cartes).

UC1 : console définie par l'utilisateur.

UR1 ou UR2 : lecteur à définir par l'utilisateur.

PTR ou PTP : lecteur ou perforateur de bandes.

UP1 ou UP2 : perforateurs à définir par l'utilisateur.

LPT : imprimante ligne.

UL1 : imprimante à définir par l'utilisateur.

Syntaxe

STAT␣D:=R/O

```
(1)  A>STAT A:=R/O

     A>STAT
     A: R/O, Space: 160k
```

(1) L'ordre protège le disque A contre l'écriture ou l'effacement.

Un ordre **STAT** simple montre que le disque est effectivement protégé. (R/O).

```
(2)  A>^C
     A>STAT
     A: R/W, Space: 160k
```

(2) Un départ à chaud (Controle C) remet le disque en position normale, la protection ne se situait donc pas sur le disque mais en mémoire vive de l'ordinateur.

Cette protection n'est pas définitive !

```
(3)  A>STAT A:=R/O
```

(3) On reprotège le disque.

```
(4)  A>ERA *.*
     ALL (Y/N)?Y
     Bdos Err On A: R/O
(5)  A>STAT
     A: R/W, Space: 160k
```

(4) Une tentative de destruction d'un ou de tous fichiers (comme ici) est refusée par la machine.

Toute tentative de ce genre replace le disque en autorisation d'écriture comme le montre le (5).

ATTENTION : pour protéger définitivement un disque, utilisez la possibilité donnée par la fenêtre de verrouillage du disque.

STAT

RENDRE UN FICHIER OPAQUE AU DIRECTORY

Syntaxe

STAT␣FICHIER.EXT_$SYS

```
(1)  A>DIR
     A: DUMP     ASM : MBASIC   COM : PIP      COM
(2)  A>STAT A:MBASIC.COM $SYS

     MBASIC.COM set to SYS
(3)  A>DIR
     A: DUMP     ASM : PIP      COM
```

(1) Un directory sur le disque A donne la liste des fichiers exitant.

(2) On passe le fichier A:MBASIC.COM en invisible.

(3) Un nouveau directory ne le laisse pas apparaître.

Cette procédure est utilisée pour mettre des fichiers en sécurité, nous verrons que cette sécurité n'est pas absolue, notamment sous un langage évolué.

Syntaxe

STAT␣FICHIER.EXT␣$DIR

```
(1)  A>DIR
     A: DUMP      ASM : PIP       COM : SUBMIT   COM
     A: ED        COM : ASM       COM : DDT      COM
     A: STAT      COM : DUMP      COM : COPYDISK COM
     A: COPYSYST  COM
(2)  A>STAT A:MBASIC.COM $DIR

     MBASIC.COM set to DIR
(3)  A>DIR
     A: DUMP      ASM : MBASIC    COM : PIP      COM
     A: XSUB      COM : ED        COM : ASM      COM
     A: LOAD      COM : STAT      COM : DUMP     COM
     A: FORMAT    COM : COPYSYST  COM
     A>_
```

(1) Un directory du disque A donne les fichiers accessibles.

(2) On place le fichier en visible, la réponse :

MBASIC.COM set to DIR

montre que le fichier existait, sinon on aurait obtenu la réponse :

No File.

(3) Un nouveau Directory laisse apparaître le fichier.

STAT

PROTEGER UN FICHIER CONTRE LA DESTRUCTION

Syntaxe

STA␣FICHIER.EXT␣$R/O

```
(1)  A>STAT XSUB.COM $R/O

     XSUB.COM set to R/O
(2)  A>ERA XSUB.COM
     Bdos Err On A: File R/O
(3)  A>ERA XSUB.COM
     Bdos Err On A: File R/O
(4)  A>^C
     A>ERA XSUB.COM
     Bdos Err On A: File R/O
     A>_
```

(1) On place le fichier **XSUB.COM** en lecture seulement, il ne pourra être ni effacé ni détruit.

(2) Une tentative d'effacement ne peut aboutir.

(3) Une nouvelle tentative n'aboutit toujours pas, donc contrairement à la protection du disque entier (**STA A:=R/O**), il n'y a pas de retour automatique en écriture.

(4) Un démarrage à chaud ne déverrouille pas davantage le fichier.

Contrairement à l'ordre de protection du disque entier qui était mémorisé en mémoire vive, l'ordre de protection d'un fichier est mémorisé sur le disque, il est définitivement acquis sauf ordre de déverrouillage explicite.

Syntaxe

STAT␣FICHIER.EXT␣$R/W

```
(1)  A>ERA XSUB.COM
     Bdos Err On A: File R/O
(2)  A>STAT XSUB.COM $R/W

     XSUB.COM set to R/W
(3)  A>DIR
     A: DUMP      ASM : MBASIC    COM : PIP       COM
     A: XSUB      COM
(4)  A>ERA XSUB.COM
(5)  A>DIR
     A: DUMP      ASM : MBASIC    COM : PIP       COM
```

(1) Une tentative de destruction de XSUB.COM n'aboutit pas, le fichier est protégé.

(2) On déverrouille le fichier.

(3) On vérifie l'existence du fichier au Directory.

(4) On tente de détruire le fichier, aucun message d'erreur n'apparaît.

(5) Le directory ne fait pas apparaître le fichier XSUB.COM qui a été détruit.

CAS PARTICULIERS DES FICHIERS OPAQUES ET PROTEGES LE BASIC

```
      A>
(1)   A>STAT A:MBASIC.COM $SYS

      MBASIC.COM set to SYS
(2)   A>STAT A:XSUB.COM $R/O

      XSUB.COM set to R/O
(3)   A>MBASIC
      BASIC-80 Rev. 5.21

      Copyright 1977-1981 (C) by Microsoft
      30455 Bytes free
      Ok
(4)   FILES
      DUMP    .ASM  MBASIC  .COM  PIP     .COM XSUB  .COM
      ED      .COM  ASM     .COM  DDT     .COM STAT  .COM
      DUMP    .COM  COPYDISK.COM  FORMAT  .COM
      Ok
```

Les fichiers opaques ou protégés apparaissent à l'ordre **FILES** du Basic sous une syntaxe particulière.

Les fichiers **opaques au Directory** sont **visibles sous Basic** mais la seconde lettre de l'extension est en inversion vidéo, ici c'est le cas de **MBASIC. COM** qui a été caché en (1) et qui apparaît en (4).

Les fichiers protégés en écriture sont visibles à l'ordre **FILES** du Basic mais la première lettre de leur extension est en inversion vidéo.

Syntaxe

STAT␣CON:=CRT:

où **CON** est un exemple de nom d'entrée/sortie et
CRT un exemple de caractéristique.

```
(1)  A>STAT DEV:
     CON:  is  TTY:
     RDR:  is  TTY:
     PUN:  is  TTY:
     LST:  is  TTY:

(2)  A>STAT RDR:=CRT:

     Invalid Assignment
(3)  A>STAT CON:=CRT:

(4)  A>STAT DEV:
     CON:  is  CRT:
     RDR:  is  TTY:
     PUN:  is  TTY:
     LST:  is  TTY:

     A>_
```

(1) On demande les caractéristiques actuelles pour les quatre périphériques.

(2) On veut assigner CRT (Grande vitesse) à RDR, la machine le refuse (voir STAT VAL:), cette caractéristique n'est pas possible.

(3) On assigne CRT à CON (clavier/écran), la machine l'accepte

(4) L'ordre **STAT DEV:** vérifie la modification de caractéristiques de CON.

Remarque : reportez-vous à l'ordre **STAT DEV:** pour la description et la signification des codes ; il faut remarquer que le plus souvent, les valeurs par défaut assignées par le constructeur donnent les meilleurs résultats en fonction des périphériques standards de la marque.

ENCHAINER PLUSIEURS COMMANDES

Syntaxe

SUBMIT PROG

```
A>ED DEPT.SUB

NEW FILE
     : *I
    1:  DIR
    2:  STAT
    3:
     : *E

A>SUBMIT DEPT

A>DIR
A: ED        COM : STAT      COM : DEPT      SUB
A: PIP       COM : SUBMIT    COM : XSUB      COM
A>STAT
A: R/W, Space: 210k


A>_
```


Sous éditeur, on crée le fichier DEPT.SUB (l'extension SUB est obligatoire). Ce fichier contient les ordres DIR puis STAT.

L'ordre **SUBMIT DEPT** lance l'exécution des ordres contenus dans le fichier DEPT donc DIR puis STAT.

Dans l'ordre **SUBMIT**, le fichier exécutable doit obligatoirement posséder l'extension SUB sinon un message d'erreur apparaitra. De ce fait, le fichier appelé par SUBMIT (ici DEPT) n'a pas à mentionner l'extension.

MISE A L'HEURE DE L'HORLOGE TEMPS REEL

CP/M 86 uniquement

Syntaxe

TOD date heure

Exemple

TOD 2Ø/Ø1/84 1Ø:15:ØØ

Visualisation et mise à jour de la date et de l'heure v1.1 utiliser une touche pour mettre à l'heure.

Le système répond une phrase qui est particulière à une machine ; ce n'est qu'au moment où l'on frappe une touche que la mise à l'heure s'effectue.

La date est rentrée en trois nombres de deux chiffres séparés par un trait de fraction (/).

L'heure est entrée en trois nombres de deux chiffres séparés par deux points.

L'heure est donnée sur un cycle de 24 heures (pas de PM ou AM)

Les deux blocs de données sont séparés par un espace.

AFFICHER L'HEURE ET LA DATE

CP/M 86 uniquement

Syntaxe

TOD

Exemple

A TOD

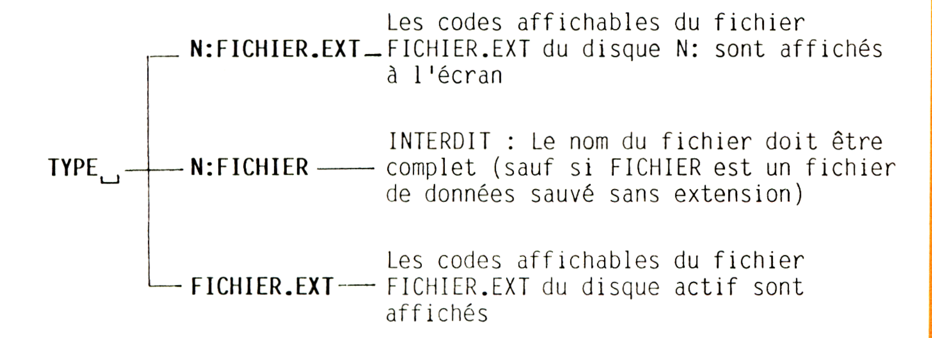
Visualisation et mise à jour de la date et de l'heure v1.1
2Ø/Ø1/84 1Ø:56:38

L'ordre **TOD** simple affiche à l'écran l'heure et la date qui sont conservés par l'horloge interne de l'ordinateur.

Remarque : certaines machines possèdent un accumulateur électrique qui conserve l'activité de l'horloge même durant les périodes où l'électricité est débranchée.

A défaut de cet équipement et si la mise à l'heure et au jour n'a pas été faite, l'ordre **TOD** donnera comme heure la durée écoulée depuis la mise sous tension et comme date une valeur variable selon la machine.

TABLEAU RECAPITULATIF : ORDRE TYPE



Règles :
- Le fichier à lister doit être caractérisé par son nom complet, extension comprise.
- Aucune ambiguïté dans le nom n'est admise.
- Seront seuls listés les signes dont les codes permettent le listage (codes ASCII).
- Les formes de stockage condensé n'apparaîtront pas.

Syntaxe

TYPE␣FICHIER.BAS

```
A>TYPE DEPART.BAS
?c2ï ?ü(
A>_
```

Le fichier DEPART.BAS a été sauvé sous la forme condensée (sous Basic, ordre **SAVE "DEPART"**). Les codes ne sont pas les codes ASCII affichables.

```
A>TYPE DEBUT.BAS
50 PRINT CHR$(26)
100 FOR I=1 TO 3:PRINT TAB(10*I) "***";:NEXT I
200 FOR I=1 TO 2:PRINT TAB(20*I+20) STRING$(13,42);:NEXT
300 PRINT TAB(12)"*" TAB(23)"*" TAB(32)"*" TAB(52)"*"
400 PRINT TAB(12)"*" TAB(24)"*" TAB(32)"*" TAB(43)"*"
500 PRINT TAB(12)"*"TAB(25)"*"TAB(32)"*"TAB(43)"******"
600 PRINT TAB(12)"*"TAB(22)"*"TAB(26)"*"TAB(3
A>_
```

Le fichier DEBUT.BAS a été sauvé sous forme ASCII, (sous Basic, ordre **SAVE "DEBUT",A**). Les codes sont les codes ASCII affichables. Une action sur la touche BREAK a arrêté le listage.

Syntaxe

TYPE␣FICHIER.ASM

```
A>TYPE DUMP.ASM
;        FILE DUMP PROGRAM, READS AN INPUT FILE
;
;        COPYRIGHT (C) 1975, 1976, 1977, 1978
;        DIGITAL RESEARCH
;        BOX 579, PACIFIC GROVE
;        CALIFORNIA, 93950
;
         ORG      100H
BDOS     EQU      0005H    ;DOS ENTRY POINT
CONS     EQU      1        ;READ CONSOLE
TYPEF    EQU      2        ;T
A>
```

Le fichier **DUMP.ASM** est listé à l'écran avec son en-tête (copyright et dates).

Le fichier sauvé en assembleur voit tous ses codes apparaître à l'écran.

Remarque : le fichier DUMP.ASM est beaucoup plus long que ce qui apparaît sur l'exemple. L'action sur la touche BREAK a arrêté le listage définitivement.

Pour arrêter momentanément le défilement du fichier, on agit sur la touche CONTROL F (Touche CONTROL appuyée puis, sans la relâcher, appuyer sur la touche F) une seconde action sur les mêmes touches provoque le redéfilement (position bistable).

TABLEAU RECAPITULATIF : ORDRE USER

USER␣N	Rend la zone utilisateur **N** active. **N** peut aller de Ø à 15 soit 16 zones possibles. Par défaut, à la mise en service, c'est la zone Ø qui est active. Pour charger des fichiers dans une zone utilisateur, il est nécessaire de suivre une procédure particulière décrite à la fiche "Utiliser les zones USER".

PASSER DANS UNE ZONE UTILISATEUR

```
A>USER 5
A>DIR
NO FILE
A>USER 0
A>DIR
A: DEBUT    BAS : DEPART   BAS : DEBUG    COM
A: STAT     COM : ED       COM
A>_
```

L'ordre **USER 5** passe l'utilisateur en zone 5 ; un ordre **DIR** montre que cette zone ne possède aucun fichier.

L'ordre **USER Ø** permet de revenir à la zone par défaut qui possède des fichiers.

TABLEAU RECAPITULATIF/EDITEUR ED
ORDRES PRINCIPAUX

nA	(Append,Apporte) Place dans la mémoire de l'éditeur n lignes du fichier édité
B	(Begin) place le pointeur de l'éditeur sur la première ligne de la mémoire d'édition.
nT	Affiche n lignes de la mémoire d'édition, le pointeur restant en place. (Texte) -nT affiche les lignes en remontant à partir du pointeur.
I	Début d'Insertion de texte au niveau du pointeur.
CTRL Z	Fin d'insertion.
nL	Avance de n Lignes dans la mémoire d'édition -nL recule de n Lignes dans la mémoire d'édition à partir du pointeur.
Ftexte	(Find) recherche le texte et place le pointeur immédiatement après.
Stexte1CTRLZ	texte 2 Recherche texte 1 et lui substitue texte 2 (Substitue).
nK	(Kill) Détruit n Lignes à partir du pointeur -nK détruit n Lignes en remontant, à partir du curseur.
nC	Déplacement de n Caractères. -nC déplace en reculant.
nD	Détruit n caractères. -nD détruit en reculant.
Q	Quitter l'éditeur sans mise à jour.
E	Quitter l'éditeur avec sauvegarde de la version mise à jour (Exit)
O	Revient en mode édition au fichier Original en ignorant les modifications que l'on vient de faire.
ØV	Donne la place restant disponible en édition par rapport à la place initialement disponible dans la mémoire de l'éditeur.

Remarque : Seules les commandes essentielles sont décrites ici. Ces commandes sont reçues par l'éditeur à la suite du message de disponibilité : *

```
A>ED TEXTE

NEW FILE
      : *I  (R)                                                   (R)
     1:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
     2:  PAR L'EDITEUR ED. (R)
     3: ↑Z
      : *B  (R)
     1: *T  (R)
     1:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
     1: *#T (R)
     1:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
     2:  PAR L'EDITEUR ED.
     1: *E  (R)

A>_
```

La réponse "NEW FILE" signale que le fichier nommé n'existe pas encore et qu'il est donc créé.

La commande **I** : indique une insertion de texte en majuscules.
i : permet l'introduction minuscules et majuscules.

L'ordre **CTRL Z** : indique la fin d'insertion.

La commande **B** : place le pointeur de travail au début de la mémoire d'édition.

La commande **T** : affiche une ligne de texte.

La commande **#T** : affiche tout le contenu de la mémoire d'éditeur.

La commande **E** : indique la fin de la séquence de travail avec sauvegarde du travail dans le fichier de nom TEXTE.

La fin de chaque ligne introduite est marquée par la touche ENVOI (Return, Enter, LF ...).

La commande **E** a pour effet de sauver le travail sur disque, il est donc nécessaire de vérifier avant de la lancer, la fermeture du lecteur et la présence de la disquette.

Les touches control sous ED

Certaines touches contrôlées ont un effet particulier sous le programme éditeur ED :

CTRL C : Abandon de l'éditeur et retour sous CP/M.
CTRL E : Retour chariot sans fin de ligne éditeur. La ligne éditeur peut comprendre plusieurs lignes physiques d'écran ou d'imprimante.
CTRL H : Détruit le dernier caractère frappé (équivaut à la touche RUB de certaines machines).
CTRL U : Détruit la ligne entière en cours de frappe.

```
A>ED TEXTE
      : *#AB#T (R)
    1:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    2:  PAR L'EDITEUR ED.
    1: *I         ECRITURE D'UN TEXTE AVEC ED     (R)
*B#T  (R)
    1:            ECRITURE D'UN TEXTE AVEC ED
    2:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    3:  PAR L'EDITEUR ED.
    1: * (R)
    2:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    2: *I         --------------------------------    (R)
*B#T (R)
    1:            ECRITURE D'UN TEXTE AVEC ED
    2:            --------------------------------
    3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:  PAR L'EDITEUR ED.
    1: *E (R)

A>DIR
A: TEXTE    BAK : TEXTE          : ED         COM
A>_
```

La commande **#A** charge dans la mémoire d'édition toutes les lignes du fichier disque TEXTE (la commande 2A aurait chargé les deux premières lignes).

La commande **B** place le pointeur au début de la mémoire.

La commande **#T** liste à l'écran la totalité de la mémoire (la commande 2T aurait listé 2 lignes à partir du pointeur ; la commande -2T aurait listé deux lignes à partir du pointeur, en remontant ; ici, le pointeur étant au début, cette commande n'aurait eu aucun effet).

La commande **I** permet d'insérer une ligne, l'envoi (R) indique la fin d'insertion.

B#T permet de vérifier l'insertion.

Une seconde ligne est insérée, le listage vérifie l'existence de 4 lignes de texte.

E termine l'édition et sauve sous le nom TEXTE le fichier modifié.

L'ordre **CP/M DIR** permet de voir que l'ancien fichier est conservé sous l'extension .BAK (BACKUP).

INSERTION D'UN TEXTE AU DEBUT D'UNE LIGNE

```
A>ED TEXTE
      : *#AB#T  (R)
    1:              ECRITURE D'UN TEXTE AVEC ED
    2:              --------------------------------------------
    3:   CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:   PAR L'EDITEUR ED.
    1: *3L  (R)
    4: *IPREFERE ^Z    (R)

 *B#T  (R)


    1:              ECRITURE D'UN TEXTE AVEC ED
    2:              --------------------------------------------
    3:   CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:   PREFERE PAR L'EDITEUR ED.
    1: *E  (R)

A>_
```

La séquence **#AB#T** charge le fichier édité dans la mémoire, place le pointeur au début et liste la totalité du texte.

Les ordres d'édition peuvent être groupés :

- La commande **3L** place le pointeur 3 lignes plus loin.
- La commande **IPREFERE CTRLZ** insère le texte PREFERE à partir de la position du curseur (I=début d'insertion, CTRLZ=fin).
- La commande **B#T** permet de lister toute la mémoire depuis le début.

E termine la séquence en sauvant sur disque la nouvelle version.

```
A>ED TEXTE
   : *#AB#T (R)
  1:            ECRITURE D'UN TEXTE AVEC ED
  2:            ------------------------------
  3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
  4:  PREFERE PAR L'EDITEUR ED.
  1: *-BI  (R)
  5:  L'EDITEUR PERMET DE SUPPRIMER FACILEMENT UNE LIGNE
  6:  DE TEXTE
  7: ↑Z
   : *-2L  (R)
  5: *K    (R)
  5: *B#T  (R)
  1:            ECRITURE D'UN TEXTE AVEC ED
  2:            ------------------------------
  3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
  4:  PREFERE PAR L'EDITEUR ED.
  5:  DE TEXTE
  1: *4LK  (R)
   : *Q    (R)

Q-(Y/N)?Y

A>_
```

L'ordre **-BI** cumule le passage du pointeur en fin de mémoire (-B) et l'ordre d'insertion début (I). A la fin du texte inséré, on termine par CTRL Z.

L'ordre **-2L** fait remonter le curseur de 2 lignes (7-2=5).

L'ordre **K** détruit la ligne ce qui est vérifié par la séquence B#T qui liste toute la mémoire.

La séquence 4LK fait avancer le pointeur de 4 Lignes (1+4=5) et détruit la ligne (donc la 5).

L'ordre **Q** permet de quitter l'édition sans sauver la version de travail (ici le texte de fin était le même que celui de début donc inutile de le sauver).

```
A>TYPE TEXTE
         ECRITURE D'UN TEXTE AVEC ED
         ------------------------------
CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
PREFERE PAR L'EDITEUR ED.

A>_
```

Sous CP/M, l'ordre **TYPE** permet de lister en clair un fichier créé sous éditeur ED.

EDITEUR

INSERER DU TEXTE A L'INTERIEUR D'UNE LIGNE

```
A>ED TEXTE
     : *#AB#T (R)
    1:             ECRITURE D'UN TEXTE AVEC ED
    2:             ------------------------------------
    3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:  PREFERE PAR L'EDITEUR ED.
    1: *3L   (R)
    4: *FEDITEUR  (R)
    4: *I DE TEXTE^Z
*B#T  (R)
    1:             ECRITURE D'UN TEXTE AVEC ED
    2:             ------------------------------------
    3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:  PREFERE PAR L'EDITEUR DE TEXTE ED.
    1: *E  (R)

A>_
```

Passons une dernière fois sur la séquence #AB#T qui est une séquence systématique au début de tout travail sous ED.

L'ordre **3L** place le pointeur 3 lignes plus loin donc ligne 4.

L'ordre **FEDITEUR** (pour Find EDITEUR CaD Cherche le mot EDITEUR) place le pointeur immédiatement APRES le mot. Sans l'ordre 3L, le pointeur se serait placé au même endroit puisque le mot ne se trouve pas dans les lignes précédentes.

L'ordre **I DE TEXTECTRLZ** permet d'insérer le texte qui est entre les marques de début et de fin d'insertion (I et CTRLZ) donc les mots DE TEXTE.

L'ordre **B#T** permet de vérifier à l'écran le contenu de la mémoire de travail et donc si l'insertion a bien été faite.

```
A>ED TEXTE
     : *#AB#T
    1:           ECRITURE D'UN TEXTE AVEC ED
    2:           ---------------------------
    3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:  PREFERE PAR L'EDITEUR DE TEXTE ED.
    1: *SL'EDITEUR^ZLE PROGRAMME D'EDITION   (R)
    4: *B#T (R)
    1:           ECRITURE D'UN TEXTE AVEC ED
    2:           ---------------------------
    3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
    4:  PREFERE PAR LE PROGRAMME D'EDITION DE TEXTE ED.
    1: *E  (R)

A>_
```

L'ordre **SL'EDITEURCTRLZLE PROGRAMME D'EDITION** signifie échanger (Substitute) à la suite de mots "L'EDITEUR" la suite de mots "LE PROGRAMME D'EDITION".

L'ordre **B#T** permet de lister toute la mémoire depuis le début et donc de vérifier que la substitution s'est correctement effectuée.

On remarque qu'une recherche ou une substitution peut se faire quelle que soit la position du pointeur pourvu qu'il soit placé en avant du mot à rechercher ou à substituer.

La substitution faite, le pointeur se trouve sur la ligne et à la fin de la chaîne substituée.

L'ordre **E** sauve la version modifiée du texte.

DEPLACER LE CURSEUR SUR UNE LIGNE
DETRUIRE DES CARACTERES

```
A>ED TEXTE
      : *#AB#T
     1:             ECRITURE D'UN TEXTE AVEC ED
     2:             --------------------------------
     3:  CECI EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
     4:  PREFERE PAR LE PROGRAMME D'EDITION DE TEXTE ED.
     1: *2L  (R)
     3: *5C  (R)
     3: *T   (R)
EST UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
     3: *4C  (R)
     3: *T   (R)
UN TEXTE INTRODUIT DANS VOTRE ORDINATEUR
     3: *9D  (R)
     3: *B#T (R)
     1:             ECRITURE D'UN TEXTE AVEC ED
     2:             --------------------------------
     3:  CECI EST INTRODUIT DANS VOTRE ORDINATEUR
     4:  PREFERE PAR LE PROGRAMME D'EDITION DE TEXTE ED.
     1: *E   (R)

A>_
```

L'ordre **2L** pousse le pointeur de 2 lignes.

L'ordre **5C** pousse le pointeur de 5 caractères (il était en début de ligne : il passe donc au 5e caractère, l'espace entre CECI et EST :

```
↓               ↓              ↓
CECI EST  (5C)  CECI EST  (4C) CECI EST UN

 avant         1° opération    2° opération
```

L'ordre **T** liste la mémoire à partir de la position du pointeur.

L'ordre **9D** détruit 9 caractères à partir du pointeur.

La séquence **B#T** permet de lister la totalité de la mémoire modifiée.

E sauve sur disque la version modifiée.

TOURS DE MAIN

Les pages qui vont suivre ont pour objectif, maintenant que vous êtes familiarisé avec CP/M, de vous donner quelques astuces de praticien.

Ces tours de main, fruit d'expériences, ne sont pas toujours adaptés à certaines situations, ils ont le mérite d'être efficaces et directement réalisables par tout possesseur de CP/M.

Certains thèmes sont liés à la sécurité des systèmes ou à celle des disquettes, nous n'insisterons jamais assez sur l'aspect critique des supports magnétiques, une très grande rigueur est nécessaire dans les procédures de manipulation et de sauvegarde des données pour ne pas prendre de risque de détérioration.

Il peut être nécessaire, dans certains cas de faire appel aux premières pages de ce livre pour retrouver la signification de certains termes.

Le système CP/M est un système qui offre tous les outils y compris ceux nécessaires à la programmation en assembleur.

Nous avons pris le parti d'un livre simple dont la particularité est la facilité d'accès à une information et non l'exhaustivité par rapport au système CP/M.

Nous ne ferons donc que citer pour mémoire les commandes disponibles qui sont :

ASM: ou ASM 86 est le programme assembleur proprement dit ; il est lié à la mnémonique du microprocesseur 8Ø8Ø d'Intel. Si vous voulez travailler en assembleur avec une machine équipée du micro processeur Z8Ø, il faudra acquérir le programme ZSID adapté à votre machine.

GENCMD: en CP/M 86, c'est l'utilitaire qui génèrera, à partir du fichier hexa écrit par le programme ASM 86 le fichier en code exécutable par le processeur.

SAVE: permet de stocker sur disque le programme assemblé, on l'utilisera aussi pour sauver des programmes modifiés.

Si vous souhaitez aller plus loin dans cette direction, nous vous suggérons les ouvrages suivants :

- CP/M pas à pas (A.PINAUD) Editions du PSI.
- Programmation du Z 8Ø (R. ZAKS) Editions Sybex.
- Programmer en Assembleur (A.PINAUD) Editions du PSI.

Cet outil de mise au point (**D**ynamic **D**ebugging **T**ool) est livré en standard avec CP/M.

L'objet de notre livre étant surtout orienté vers la pratique de tous les jours, nous n'insisterons pas sur l'utilisation d'un outil de mise au point.

Le chapitre qui va suivre sera simplement un rappel des ordres de fonctionnement de **DDT.**

Il est important de signaler que **DDT** est un outil adapté au microprocesseur 8Ø8Ø. Si vous possédez un ordinateur construit autour d'un Z 8Ø, le ZSID.

RESUME DES COMMANDES DDT

Lancer **DDT**

Syntaxe

DDT nom de fichier

A cet ordre, le programme **DDT** proprement dit se charge à la place de CCP cependant que le programme à travailler se charge en TPA. Vous ne pourrez à partir de ce moment que travailler sous les commandes propres de DDT puis sauver vos modifications par l'ordre SAVE et revenir à CP/M pour recharger le CCP.

Exemple

```
A>DDT PIP.COM
DDT VERS 2.2
NEXT  PC
1E00 0100
-
-H1E00,0100
1F00 1D00
-
```

Les valeurs données le sont en hexadécimal.

La valeur sous le mot NEXT donne la dernière adresse utilisée par le chargement du programme de travail cependant que celle placée sous PC en donne la première ; la différence entre les deux donne l'importance du programme et donc la taille qui devra être sauvée par SAVE après modifications.

Ici, dans l'exemple précédent, l'adresse NEXT est 1EØØ, l'adresse PC est Ø1ØØ ; dans la table de conversion, on voit que

1E = 3Ø en décimal, Ø1 à la même valeur, le programme PIP.COM occupe donc 29 pages de mémoire soit 29 blocs de 256 octets.

Sous **DDT**, on peut demander à la machine de faire le calcul de somme et de différence entre deux valeurs hexadécimales par la commande :

Hvaleur,valeur

La réponse est la somme puis la différence en hexadécimal, bien sûr.

Commandes de lecture

D

Lire en hexadécimal : commande **D** (pour **DUMP**)

Syntaxe

Ddébut,fin

La commande D permet d'afficher le contenu de la mémoire sous forme d'hexadécimal. Les paramètres début et fin sont optionnels, par défaut, le dump se fera de la première adresse et sur 12 lignes, s'il n'y a que le premier paramètre, le dump se fera depuis cette adresse et sur les 12 lignes suivantes, s'il y a les deux, le dump se fait sans discontinuer de l'adresse spécifiée à l'adresse spécifiée.

L'interprétation faite dans la partie droite de l'écran n'est représentative que pour les codes ASCII, c'est-à-dire que le programme interprète chaque code comme sa valeur ASCII ; pour certains, non représentatifs, il n'y aura pas de signes écrits mais dans certains cas, il peut y avoir interprétation pour des codes compressés, donc ou significatifs.

Exemple

```
-D0200,0220
0200 20 20 20 43 4F 50 59 52 49 47 48 54 20 28 43 29     COPYRIGHT (C)
0210 20 31 39 37 39 2C 20 44 49 47 49 54 41 4C 20 52  1979, DIGITAL R
0220 45 E
-
```

L

Lire les instructions machine : commande **L** (pour **Liste**).

Syntaxe

Ldébut,fin

La commande L permet de visualiser la liste désassemblée des instructions machine selon la mnémonique 8Ø8Ø.

Les paramètres de début et de fin sont optionnels, en leur absence, il y aura 11 lignes assemblées à partir de la première adresse. Le rôle des paramètres est le même que pour le Dump.

C'est à partir de cette liste de référence que nous pourrons entre autres opérations, modifier le contenu d'une adresse en utilisant la commande de substitution.

Exemple

```
-L0200,020A
  0200  ??=  20
  0201  ??=  20
  0202  ??=  20
  0203  MOV  B,E
  0204  MOV  C,A
  0205  MOV  D,B
  0206  MOV  E,C
  0207  MOV  D,D
  0208  MOV  C,C
  0209  MOV  B,A
  020A  MOV  C,B
  020B
-_
```

Commandes de modification de la mémoire

S

Modifier le contenu d'une mémoire : commande S (pour **Substituer**).

Syntaxe

Sadresse

La commande S permet de remplacer la valeur hexadécimale affichée par celle que l'on frappe.

La commande se reproduit pour l'adresse suivante automatiquement, la commande de substitution est donc automatiquement répétitive. La frappe d'un point ramène à DDT (La commande CTRL Z produirait le même effet).

Exemple

```
-D0200,020E
0200 20 20 20 43 4F 50 59 52 49 47 48 54 20 28 43     COPYRIGHT (C
-
-S0201
0201 20 41
0202 20 .
-
-D0200,020E
0200 20 41 20 43 4F 50 59 52 49 47 48 54 20 28 43   A COPYRIGHT (C
-
-
```

Le code 41 (A majuscule) a remplacé le code 2Ø.

Commandes de lecture des registres du processeur

Lister le contenu des registres à un moment.

Syntaxe

X

La commande X affiche le contenu des registres utilisés par le microprocesseur.

Cette commande ne vous sera d'aucune utilité si vous ne connaissez pas le fonctionnement du microprocesseur qui équipe votre machine, cette remarque est à étendre à la commande T pour trace qui, instruction par instruction donne l'état de ces mêmes registres.

Syntaxe

T valeur hexa

La valeur Hexa est le nombre d'instructions que la machine doit opérer en donnant après chacune d'elle le contenu des registres.

Exemple

```
-
-X
C0Z0M0E0I0 A=00 B=0080 D=0000 H=0000 S=1DF0 P=04D5 MVI  E,80
-
-T
C0Z0M0E0I0 A=00 B=0080 D=0000 H=0000 S=1DF0 P=04D5 MVI  E,80*04D7
-
-T3
C0Z0M0E0I0 A=00 B=0080 D=0080 H=0000 S=1DF0 P=04D7 LXI  B,1ECC
C0Z0M0E0I0 A=00 B=1ECC D=0080 H=0000 S=1DF0 P=04DA CALL 0A18
C0Z0M0E0I0 A=00 B=1ECC D=0080 H=0000 S=1DEE P=0A18 LXI  H,1F77*0A1B
-_
```

G

Brancher le programme sur une adresse.

Syntaxe

Gadresse

Où adresse est l'adresse hexadécimale où doit démarrer le programme.

Cette commande est entre autre typiquement utilisée pour terminer un travail de mise au point sous la forme : GØ qui branche le programme sur l'adresse Ø donc au début de la mémoire : ceci équivaut à sortir de DDT et à revenir à CP/M.

Nous avons vu les principales commandes de DDT, reportez-vous à la notice constructeur pour quelques autres qui revêtent un intérêt moins évident.

```
-S0201
0201 41 20
0202 20 .
-
-D0200,020E
0200 20 20 20 43 4F 50 59 52 49 47 48 54 20 28 43     COPYRIGHT (C
-
-G0

A>_
```

6Ø en dernière ligne rend la main à CP/M.

Exemple d'une séquence de travail sous DDT

```
A>DDT PRINTER.COM
DDT VERS 2.2
NEXT  PC
0580 0100
-
-H0580,0100
0680 0480
-
-D0200,022E
0200 50 42 00 00 ED 73 48 FD 31 6A FD E5 D5 C5 79 CD PB...sH.1j....y.
0210 A1 FA C1 D1 E1 ED 7B 48 FD C9 3E 00 CD A1 FA 3E ......{H..>....>
0220 0A FE 1F C8 21 B0 FC FE 09 28 54 FE 0C 20 17 ....!....(T.. .
-
-S0200
0200 50 FF
0201 42 00
0202 00 .
-
-D0200,022E
0200 FF 00 00 00 ED 73 48 FD 31 6A FD E5 D5 C5 79 CD .....sH.1j....y.
0210 A1 FA C1 D1 E1 ED 7B 48 FD C9 3E 00 CD A1 FA 3E ......{H..>....>
0220 0A FE 1F C8 21 B0 FC FE 09 28 54 FE 0C 20 17 ....!....(T.. .
-
 -
```

```
-G0

A>SAVE 5 PRINT1.COM
A>
A>DDT PRINT1.COM
DDT VERS 2.2
NEXT  PC
0600 0100
-
-D0200,020E
0200 FF 00 00 00 ED 73 48 FD 31 6A 1D E5 D5 C5 79 ......sH.1j...../
-G0

A>_
```

L'opération consiste à changer le nombre de caractères par défaut dans une commande d'imprimante : nous voulons le porter à 255 (FF en hexadécimal) pour qu'un programme de traitement de texte dirige le passage à la ligne, même au-delà de 5Ø caractères.

1 - On charge le programme de référence (PRINTER,OM) sous DDT.

2 - La commande H vous donne l'encombrement du fichier (Ø48Ø donc un peu moins de 5 pages de 256 octets, exactement 4,8).

3 - Un DUMP du début de programme nous donne la valeur par défaut à l'adresse Ø2ØØ : c'est la documentation du constructeur qui indique cette position.

4 - On substitue FF à 5Ø à l'adresse 2ØØ. On place à l'adresse 2Ø1, c'est la case du nombre de lignes par page, le point de l'adresse 2Ø2 nous permet de sortir du mode substitution.

5 - Un DUMP des lignes nous permet de vérifier.

6 - GØ nous ramène à CP/M.

7 - L'ordre SAVE crée un nouveau fichier (PRINT 1.COM) de 5 pages de long (voir 2) qui est le programme de référence modifié.

8 - Nous vérifions que le programme sauvé est bien le programme modifié.

CP/M 86 uniquement

L'outil de mise au point **DDT** qui existe en CP/M a son équivalent en CP/M 86. Alors que **DDT** a une mnémonique liée au microprocesseur 8Ø8Ø ce qui le rend relativement inefficace avec les Z 8Ø plus répandus, le DDT 86 a une mnémonique 8Ø86 et 8Ø88, donc adaptée aux microprocesseurs les plus fréquents.

Syntaxe

DDT86 nom de fichier

RESUME DES COMMANDES

Commandes

Arguments des ordres :

a1 : arrêt 1
a2 : arrêt 2
d : adresse de début
d1 : seconde adresse de début
de : destination des données
f : adresse de fin
ms : mot spécifié
n : nombre d'instructions à exécuter
os : octet spécifié
r : registre du processeur
W : mot de 16 bits

Commandes

Les arguments entre parenthèses sont optionnels.

Les ordres sont en majuscules, les arguments en minuscules.

Ad	: Débute l'assemblage.
Bd,f,dl	: Compare les blocs de mémoire.
D(W)(s(,f))	: Liste la mémoire en Hexa et en ASCII
Enom de fichier	: Charge le programme spécifié pour exécution.
Fd,f,os	: Remplit la zone mémoire spécifiée avec l'octet spécifié

FWd,f,ms : Remplit la zone mémoire avec le mot de 16 bits spécifié.

G(d)(,a1(,a2)) : Débute l'exécution.

Hms1,ms2 : Donne la somme et la différence entre les deux valeurs spécifiées.

L-d(,f)) : Liste la mémoire sous forme mnémonique 8Ø86/8Ø88.

Md,f,de : Déplace le bloc mémoire spécifié.

Rnom de fichier : Lit le fichier disque dans la mémoire.

T(n) : Débute la trace de l'exécution.

TS(n) : Idem mais avec visualisation des registres du processeur en plus.

U(n) : Annule les options de trace.

V : Vérification de la mémoire après chargement depuis le disque.

Wnom de fichier(,d,f) : Ecrit sur disque, sous le nom spécifié, les blocs mémoire.

X(r) : Examine pour modification les registres du processeur.

Courrier automatique

Ce petit tour de main vous permet de créer une lettre personnalisée à partir d'une lettre standard.

Le procédé fait appel aux possibilités de l'éditeur ED livré en série avec le système d'exploitation CP/M.

Nous utiliserons les possibilités de création d'un texte, puis éventuellement celles de correction mais surtout la substitution d'une chaîne par une autre que permet l'éditeur grâce à sa commande S.

Le principe sera donc de créer une lettre dont les valeurs susceptibles de changement seront précisées à leur place dans le texte, la personnalisation du courrier n'étant que la substitution de ces chaînes de caractères par celles souhaitées.

Nous utiliserons pour finir la particularité de ED de conserver la version originale du texte sur disque pour la renommer après le travail et à ne conserver que la version de base réutilisable pour un autre courrier.

1 - Créer le texte de la lettre

```
A>ED B:LETTRE

NEW FILE
     : *i
 1:
 2:
 3:
 4:
 5:                                        Paris le DAT1
 6:
 7: Société DUPONT
 8: 17, rue des Roses
 9: 75025 PARIS
10:
11:                                        à NOM1
12:                                          ADR1
13:                                          COVI1
14:
15:
16:                              NOM1 ,
17:
18:         Nous avons bien reçu votre récent courrier
19: et accusons réception de votre commande.
20:
21:         Afin de réduire votre attente, nous vous
22: ,serions reconnaissant de bien vouloir nous adresser
23: la somme de SOM1 correspondant aux frais de port et
24: au montant de votre commande.
25:
26:         Dans cette attente, nous vous prions de croire,
27: NOM1, en nos sincères salutations.
28:
29:
30:
31:                                        Le-service commandes.
32: ↑Z
33: *E
```

Remarques : le signe d'insertion est un i minuscule ce qui permet d'écrire en minuscules ou majuscules. Si on met le I majuscule, toutes les minuscules frappées seront changées en majuscules.

La ligne 32 est le control Z de fin de fichier.

L'ordre E de la ligne 33 sauve le fichier et clot l'introduction.

2 - Modifier automatiquement les paramètres

```
A>DIR B:
B: LETTRE
A>ED B:LETTRE
      : *#A
     1: *SDAT1^Z10/10/82
     5: *#SNOM1^ZMonsieur DUMUR

BREAK "#" AT R
   27: *B
    1: *SADR1^Z25, rue Bleue
   12: *SCOVI1^Z17216  La PLAINE
   13: *SSOM1^Z165.50 F
   23: *E

A>_
```

L'ordre #A charge la lettre en mémoire.

La ligne 1 remplace DAT1 par 10/10/82 (notez la place de control Z qui finit la première chaîne).

En ligne 5, le signe # avant SNOM1 généralise la substitution qui suit à toutes les chaînes NOM1 rencontrées ; de ce fait, une fois la 3e substitution faite, la machine indique qu'elle est arrêtée en ligne 27.

L'ordre B remonte le pointeur au début du fichier pour la suite des substitutions.

La ligne 23 termine cette séquence.

On se trouve maintenant avec une lettre personnalisée qui est rangée dans le lecteur B sous le nom de "LETTRE".

L'éditeur a conservé la version précédente (non personnalisée) sous le nom "LETTRE.BAK".

Les numéros de ligne sont le témoin de la substitution, par exemple, la substitution de la ligne 1 place en début de seconde ligne le nombre 5 indiquant que la dernière substitution s'est faite ligne 5.

3 - Taper la lettre

```
TYPE B:LETTRE




                                   Paris le 10/10/82

Société DUPONT
17, rue des Roses
75025 PARIS

                                   à MONSIEUR DUMUR
                                     25, RUE BLEUE
                                     17216  LA PLAINE


                            MONSIEUR DUMUR ,

        Nous avons bien reçu votre récent courrier
et accusons réception de votre commande.

        Afin de réduire votre attente, nous vous
serions reconnaissant de bien vouloir nous adresser
la somme de 165.50 F correspondant aux frais de port et
au montant de votre commande.

        Dans cette attente, nous vous prions de croire,
MONSIEUR DUMUR, en nos sincères salutations.




                                   Le service commandes.
```

Notre ordinateur permettant la double sortie (vers l'imprimante en même temps que l'écran), l'ordre **TYPE** suffit, sinon il faut auparavant taper l'ordre Control P qui fera le même effet.

4 - Pour finir

```
A>
A>ERA B:LETTRE
A>REN B:LETTRE=B:LETTRE.BAK
A>_
```

Cette séquence permet de détruire la lettre personnalisée et de renommer la version standard (nommée avec l'extension.BAK par ED) du nom d'origine (LETTRE).

Ceci permet donc une autre utilisation pour une nouvelle lettre.

```
A>ED B:LETTRE
      : *i
     1: ↑z
      : *B
```

Vous avez remarqué que tous les mots remplacés sont en majuscules, ED se met en effet en insertion majuscules si rien ne lui a été précisé.

La séquence proposée ici, placée après la ligne #A met donc en insertion minuscules ou majuscules.

SE SERVIR DES ZONES UTILISATEUR

La version 2.2 de CP/M permet à l'aide de l'ordre **USER**, de travailler dans des zones utilisateurs différentes (les zones de Ø à 15).

A la mise sous tension, c'est la zone Ø qui est active.

Le déplacement dans une autre zone est inefficace puisque aucun programme n'y réside.

Pour utiliser une autre zone, il est indispensable d'y transférer d'abord le fichier PIC.COM, qui grâce à son paramètre G permet de transférer des programmes d'une zone à l'autre.

Il est important de connaître les avantages et les inconvénients des zones utilisateurs :

- Les zones ne sont pas protégées par un mot code, l'ordre **STAT USR:** permet de vérifier s'il y a des zones actives autre que la zone Ø.

- Des fichiers critiques sont évidemment à l'abri s'ils sont placés dans une zone autre que la zone Ø, seule une activation volontaire de la zone où ils se trouvent permet d'y accéder.

- Travailler sous Basic oblige à stocker le langage dans la zone utilisée, d'où la diminution des capacités du disque encombré d'autant de fois le langage qu'il y a de zones actives, ceci étant bien sûr vrai, quel que soit le langage évolué.

Pour transférer **PIP** dans une zone utilisateur, on utilisera l'ordre **DDT** qui charge en mémoire centrale le fichier en donnant son encombrement, puis **SAVE** qui le stockera sur disque dans la zone active.

1 - L'état des zones

```
A>DIR
A: PIP        COM : MBASIC   COM : DDT     COM
 : STAT       COM
A>USER 5
A>DIR
NO FILE
A>USER 0
A>DIR
A: PIP        COM : MBASIC   COM : DDT     COM
 : STAT       COM
A>STAT USR:

Active User : 0
Active Files: 0
A>_
```

On vérifie que la zone Ø possède 4 fichiers.

La zone 5 ne possède aucun fichier.

La zone Ø est bien la zone active par défaut.

L'ordre **STAT USR:** confirme que la zone Ø est active et qu'elle est la seule à avoir été utilisée dans le disque.

2- Rendre la zone 5 active

```
A>DDT PIP.COM
DDT VERS 2.2
NEXT  PC
1E00 0100
-G0

A>USER 5
A>SAVE 30 PIP.COM
A>DIR
A: PIP        COM
A>USER 0
A>STAT USR:

Activè User : 0
Active Files: 0 5
A>_
```

L'appel sous **DDT** du fichier PIP.COM charge ce fichier en mémoire centrale et nous donne l'encombrement en blocs de 256 octets (les pages).

PIP occupe 1E en hexadécimal soit 3Ø pages (voir le tableau de conversion décimal-hexadécimal).

On repasse sous CP/M par l'ordre **GO** (Go=aller en Ø).

On passe en zone utilisateur 5 (**USER 5**).

On sauve les 3Ø pages de PIP (**SAVE 3Ø PIP.COM**).

L'ordre DIR laisse apparaître la présence de PIP.COM dans la zone 5.

Si on repasse en zone Ø et que l'on demande les statistiques user (STAT USR:), on vérifie que les 2 zones sont maintenant actives ; la zone Ø et la zone 5.

```
A>USER 5
A>PIP MBASIC.COM=MBASIC.COM[G0]

A>DIR
A: PIP       COM : MBASIC   COM
A>_
```

Un nouveau passage en zone 5 permet maintenant, grâce à l'ordre **PIP** associé à son argument G, d'appeler des programmes d'autres zones.

L'ordre **PIP MBASIC.COM=MBASIC.COM|GO|** copie dans la zone active (la 5) le fichier issu de la zone Ø (GØ).

L'ordre **DIR** vérifie le transfert.

Mise au point d'un programme

La possibilité de référence ambigüe d'un nom de fichier permet facilement de conserver les versions successives de la mise au point d'un programme.

Pour des raisons de sécurité, il est indispensable, dès le début du travail de frappe du programme, de sauver régulièrement, au moins toutes les demi-heures, les versions du programme.

Si le programme n'est pas trop long, on peut sauver les versions avec une extension donnant la version :

- On sauve d'abord **PROG.ØØ1**, puis **PROG.ØØ2** puis **PROG.ØØ3 ...**
- Lors de la mise au point définitive, on sauve la dernière version sous un autre nom par exemple **PROGRAM.BAS** et on détruit toutes les versions de travail avec l'ordre **CP/M ERA PROG.*** qui efface toutes les versions successives de PROG.

Si le programme est long, on a intérêt à sauver des versions cycliques pour conserver de la place sur le disque :

- On sauve d'abord **PROG.ØØ1** puis **PROG.ØØ2** puis **PROG.ØØ3** puis à nouveau **PROG.ØØ1** qui se substitue à la première version... puis ØØ2, ØØ3 et à nouveau ØØ1...

On a donc un historique sur les trois dernières versions sans trop encombrer l'espace disque.

A la fin de la mise au point du programme, on procède comme plus haut :

On sauve la version définitive sous un autre nom par exemple **PROGRAM.BAS** et on détruit les versions de travail par l'ordre **ERA PROG.***

Selon les caractéristiques électriques du lieu où vous travaillez, notamment dans des endroits où les coupures de courant sont fréquentes (orages par exemple), il est utile de sauver de temps en temps une version sur un autre disque après avoir fait son redémarrage à chaud. (voir "Changer un disque").

Cette sécurité, liée à l'instabilité électrique, est aussi à appliquer dans le cas de fichiers de données.

CHANGER UN DISQUE DANS UN LECTEUR

Voici une opération apparemment fort simple or, elle est à l'origine du plus grand nombre de déboires des utilisateurs débutants de CP/M.

Lors de la mise en service d'une disquette, CP/M charge la configuration de celle-ci en mémoire vive pour pouvoir accéder plus rapidement aux informations. Cependant, avant toute écriture, la machine vérifie l'accord entre disquette et contenu en mémoire vive grâce à une clé. Si, il y a incohérence entre la clé de la disquette et celle de la mémoire (cas de changement de disquette), il y a refus d'écriture. Il faut donc après tout changement accorder mémoire et disquette ; c'est le rôle du "démarrage à chaud".

Sous CP/M

Après le changement de disquette, taper **CONTROL C** (ou la touche BREAK).

La disquette tourne et l'initialisation se fait.

Sous MBASIC

Après tout changement de disquette, faire l'ordre **RESET** qui effectue l'initialisation.

Si vous tentez d'écrire sans avoir initialisé, vous obtenez un refus d'écriture et le message

BDOS ERROR IN A, READ ONLY

où **A** est le numéro du lecteur en faute.

Les fichiers de données sont un produit fragile et pourtant fondamental, il n'est pas possible d'envisager la possibilité de perdre une disquette contenant les 1ØØØ noms de vos clients par exemple.

La meilleure sécurité est la triple sauvegarde qui consiste à travailler sur trois disquettes selon le schéma suivant, les séquences pouvant être d'une heure jusqu'à une journée selon les risques que l'on accepte de prendre.

Séquence	*Travail*	*Copie*
1	1	2
2	2	3
3	3	1
4	1	2
...		

Le principe est donc de travailler (introduire de nouvelles données ou mise-à-jour) sur la sauvegarde de la séquence précédente ; un incident n'affectant au maximum que le travail d'une séquence.

Il est souhaitable de faire régulièrement une sauvegarde supplémentaire qui sera conservée en un autre lieu que celui de travail.

Un cahier des sauvegardes, avec trois colonnes :

- séquence (qui portera la date),
- numéros de disquette de travail,
- numéros de disquette de sauvegarde,

semble indispensable.

Les ennemis de votre ordinateur

- **Le tabac** : la fumée, passant souvent en force sur les disques par la ventilation, encrasse à la longue les lecteurs, d'autre part, les cendres se déposent sur les disquettes et constituent de parfaits abrasifs.
- **Les liquides** (café, sodas...) mettent en péril les claviers et les machines s'ils sont renversés accidentellement.

Les ennemis de vos disquettes

- **La moquette** : source d'électricité statique, elle peut être à l'origine de la destruction d'une disquette.
- **La poussière** : elle se dépose sur la fenêtre de lecture des disquettes et devient un abrasif lors de la mise en rotation. Ne laissez jamais une disquette hors de sa pochette lorsqu'elle est sortie du lecteur.
- **Les sources magnétiques** (aimants, électro-aimants...) Eloignez les disquettes des masses magnétiques ou métalliques (clés sur les disquettes !).
- **Le courrier** : les machines à indexer des PTT détruisent les disquettes ; expédiez celles-ci dans leur pochette entourée de papier d'aluminium et placez-les dans des enveloppes matelassées pour qu'elles ne passent pas dans les machines à indexer.
- **Les doigts sur la fenêtre de lecture** qui provoquent des dépôts et peuvent détruire les informations.

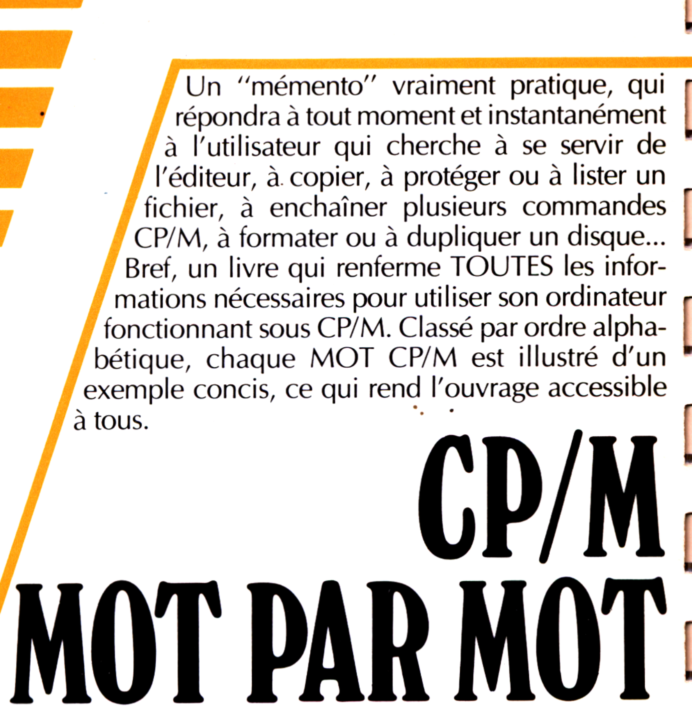
EN BREF, SUPPORT FRAGILE = PRECAUTIONS

HEXA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Achevé d'imprimer en mars 1984
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : mars 1984

N° d'impression : 305028
N° d'édition : 86595-72-2
ISBN : 2-86595-072-7

mémento



Un "mémento" vraiment pratique, qui répondra à tout moment et instantanément à l'utilisateur qui cherche à se servir de l'éditeur, à copier, à protéger ou à lister un fichier, à enchaîner plusieurs commandes CP/M, à formater ou à dupliquer un disque... Bref, un livre qui renferme TOUTES les informations nécessaires pour utiliser son ordinateur fonctionnant sous CP/M. Classé par ordre alphabétique, chaque MOT CP/M est illustré d'un exemple concis, ce qui rend l'ouvrage accessible à tous.

CP/M MOT PAR MOT

Editions du P.S.I.
B.P. 86
77400 Lagny/Marne
France

ISBN 2.86595.072.7

90 FF



Couverture Mariane de Nayer

Imprimé en France ✱✱

CP/M MOT PAR MOT

P.S.I.

Yvon Dargery



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/